AFRL-HE-WP-TR-2005-0062

AIR FORCE RESEARCH LABORATORY

Adaptive Levels of Autonomy (ALOA) for UAV Supervisory Control

Rubin Johnson
Michael Leen
Dan Goldberg
Michael Chiu

OR Concepts Applied
7032 Comstock Avenue, Suite 100
Whittier CA 90602

May 2005

Final Report for May 2004 to February 2005

Human Effectiveness Directorate
Warfighter Interface Division
Wright-Patterson AFB OH 45433

TECHNICAL REVIEW AND APPROVAL

AFRL-HE-WP-TR-2005-0062

This report has been reviewed by the Office of Public Affairs (PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public.

This technical report has been reviewed and is approved for publication.

FOR THE COMMANDER

//Signed//

MARIS M. VIKMANIS
Chief, Warfighter Interface Division
Air Force Research Laboratory

REPORT DOCUMENTATION PAGE	*Form Approved* **OMB No. 0704-0188**

1. REPORT DATE *(DD-MM-YYYY)* May 2005	**2. REPORT TYPE** Final	**3. DATES COVERED** *(From - To)* May 2004 - February 2005

4. TITLE AND SUBTITLE Adaptive Levels of Autonomy (ALOA) for UAV Supervisory Control	**5a. CONTRACT NUMBER** FA8650-04-M-6478
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER 65502F
6. AUTHOR(S) Dr. Rubin Johnson Dr. Michael Leen Dan Goldberg Michael Chiu	**5d. PROJECT NUMBER** 3005
	5e. TASK NUMBER HC
	5f. WORK UNIT NUMBER 4J
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) OR Concepts Applied 7032 Comstock Avenue, Suite 100 Whittier CA 90602	**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Air Force Research Laboratory Human Effectiveness Directorate Warfighter Interface Division Air Force Materiel Command Wright-Patterson AFB OH 45433-7022	**10. SPONSOR/MONITOR'S ACRONYM(S)** AFRL/HECI
	11. SPONSORING/MONITORING AGENCY REPORT NUMBER AFRL-HE-WP-TR-2005-0062

12. DISTRIBUTION AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

13. SUPPLEMENTARY NOTES
This is the final report for OR Concepts Applied (ORCA)'s Phase I SBIR effort entitled Adaptive Levels of Autonomy (ALOA) for UAV Supervisory Control.

14. ABSTRACT
An architecture for testing and evaluating different methods for adaptive levels of autonomy was devised. We defined multiple Levels of Autonomy (LOA) for each of four operator tasks: allocation, route planning, imagery analysis, and weapon control. To demonstrate the architecture and LOA implementation, we designed a prototype Multi-UAV Control Station Emulator research test bed, by building on existing ORCA-developed software components. ORCA's extensive internal IR&D over several years has produced state-of-the-art automated mission planning tools that allow fully autonomous execution of operator tasks. Experience with operators through the J-UCAS effort and other programs gives us first-hand knowledge of the tools and decision aids operators need when building and assessing mission plans, which support manual mission planning. With this experience, we implemented the two autonomy extremes: manual and fully autonomous and we defined and implemented intermediate levels of autonomy (requires using characteristics of both manual and autonomous task execution) for the four operator tasks noted above.

15. SUBJECT TERMS
Adaptive Autonomy, Levels of Autonomy, UAV, Autonomous Operations, Supervisory Control, Mission Planning

16. SECURITY CLASSIFICATION OF: U			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**	UU	44	Gloria Calhoun
U	U	U			**19b. TELEPONE NUMBER** *(Include area code)* 937-255-3856

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI-Std Z39-18

THIS PAGE INTENTIONALLY LEFT BLANK

Contents

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

This is the final report for OR Concepts Applied (ORCA)'s Phase I SBIR effort for AFRL/HECI entitled *Adaptive Levels of Autonomy (ALOA) for UAV Supervisory Control.* The goal of the effort is to devise an architecture for implementing and evaluating a range of adaptive levels of autonomy for UAV supervisory control and to design a Multi-UAV Control Station Emulator (MCSE) test bed to demonstrate the architecture. ORCA's Phase I successes are a direct result of years of experience designing mission planning and Low Observable (LO) autorouting tools and decision aids.

In Phase I, ORCA devised the ALOA architecture for testing and evaluating different methods for adaptive levels of autonomy. We defined multiple Levels of Autonomy (LOA) for each of four operator tasks: allocation, route planning, imagery analysis, and weapon control. To demonstrate the architecture and the implementation of the LOA, we designed a prototype Multi-UAV Control Station Emulator (MCSE) research test bed, by building on existing ORCA-developed software components. ORCA's extensive internal IR&D over several years has produced state-of-the-art automated mission planning tools that allow fully autonomous execution of operator tasks. Experience with operators through the J-UCAS effort and other programs gives us first-hand knowledge of the types of tools and decision aids operators need when building and assessing mission plans, which supports manual mission planning. This experience allows us to provide the capability to implement the two autonomy extremes: manual and fully autonomous. Implementing intermediate levels of autonomy requires using characteristics of both manual and autonomous task execution. In Phase I we defined and implemented intermediate levels of autonomy for the four operator tasks noted above. More work will be necessary to refine the LOA and to determine to the number of LOA necessary for each task.

1.0 Introduction

1.1 Envisioning Multi-Vehicle Control

Our experience with other unmanned vehicle programs, including Boeing J-UCAS, has provided us insight into what is needed for multi-UAV control and how capabilities and technologies might be employed in the future. Having a vision of what we want from a system in the future helps drive design and development. We present our vision in a scenario that shows how a multi-vehicle control station might function in future military operations.

An operator is controlling four UCAVs. The vehicles are assigned to search an area, find targets, and attack them. As the vehicles fly their preplanned search missions, the operator monitors the sensors and the health and status of the vehicles. The vehicles are equipped with automatic target recognition capability and can identify targets, but they are not authorized to shoot without the operator's consent. The UCAV team identifies two targets, and the system displays target and weaponeering data for the operator. The targets are mobile and high value. Because of the chance of escape, the strike mission must be planned and executed quickly. From training exercises and previous missions, the system has learned how to distribute workload between the operator and the computer to optimize overall system performance for this type of mission effectiveness. The system adjusts autonomy levels to facilitate quick planning. Allocation and route planning are granted a high level of autonomy. The operator is given data about the target in a customized and personalized form to suit the individual's information processing style. While the operator checks the target data--including data about the surrounding area and potential for collateral damage--the autonomous allocation and route planning modules allocate tasks to the vehicles and generate route plans. Each of the targets must be imaged to determine their locations, they must each be struck with two weapons, and finally there must be post-strike Battle Damage Assessment (BDA) of both targets. Once the missions are planned, the operator authorizes the strike. After the mission, the operator views the BDA report. Because of the importance of the targets, the operator orders two of the UCAVs to take a second look and the other two vehicles to assume a loiter pattern. The autorouter plans routes for each vehicle. As the UCAVs pass over the target area, the operator uses voice commands to slew the sensor to get a better look at the target area and confirm the kill. Satisfied that the targets have been destroyed, the operator orders the UCAV team to the next search area.

1.2 Multi-UAV Control

Although the focus of this effort is to design tools that will be used to experiment with autonomy concepts for UAV supervisory control, it is important to keep in mind the context of the problem and the larger goal: enabling multi-vehicle supervisory control. Multi-UAV supervisory control refers to a control concept in which a single operator controls a group of UAVs. In this concept, UAV flight control[1] is autonomous and the operator participates in planning, problem solving, and contingency operations (for example, a system failure). Several unmanned vehicle programs envision a future in which unmanned vehicles work together in teams and are controlled by a single operator acting in a supervisory role. The J-UCAS concept involves a single operator controlling a group of four Unmanned Combat Air Vehicles (UCAVs). The Air Force plans to use teams of Predators and armed Predator Bs to perform hunter-killer missions.[2] The Office of the Secretary of Defense UAV Roadmap (December 2002) calls for improvements in multi-vehicle supervisory control capabilities.

Increasing the capability of C2 decision aids and situation awareness tools, and implementing autonomous execution of tasks (such as target allocation and route planning) will help increase the span of control; however, more research and experimentation is required to determine the best use of these methods and tools. The current situation falls short of the goal of multi-vehicle supervisory control. While autonomous flight control is possible because it is more tractable, true multi-vehicle control is still in the conceptual stage. Some current unmanned vehicle systems require more than one operator to control a single vehicle. For example, the Navy's Tactical Control System (TCS) currently requires two operators to control a single UAV and its sensors: one operator controls and monitors the health and location of the vehicle, and the other operator manages the sensor payload and the data being transmitted back to the control station via the vehicle's sensor suite.

It is clear that if a single operator is going to control a group of UAVs, some tasks will have to be autonomous to some degree. While autonomous operations will play an important role in achieving multi-vehicle control, the human factor is critical. One obvious role for the human is to intervene in case of system failure. Another important role is for the operator to intervene when automated tools fail because of invalid modeling assumptions or algorithmic idiosyncrasies. Automated mission planning tools use underlying models of the real world and algorithms to solve problems. On rare occasions, the solutions will be suboptimal due to invalid underlying assumptions. Automated tools may also produce poor results because of bad data. In such cases, the operator must intervene to modify the answer. Making use of human experience and knowledge is an important aspect of optimizing multi-vehicle control system performance.

[1] Flight control in this context refers to the autopilot that flies the route provide by the operator.
[2] UAVS AND THE HUMAN FACTOR, J.R. Wilson, AIAA-Aerospace America Online.

To allow the operator to perform planning, monitoring, and intervention duties effectively, the system must provide the operator with situation awareness and manage the operator's workload (or permit the operator to manage workload). Situation awareness requires data, but providing too much data, or data that is difficult to understand, will diminish situation awareness. To enhance situation awareness, the right data must be provided to the operator when it is needed and in a form that is easily understandable. Exploring tools that enhance situation awareness and performance is another important dimension of this effort

1.3 Multi-Vehicle Mission Planning Capabilities

Mission planning is decision making to address air war force employment. The basic problem is to avoid threats and accomplish mission objectives. There are several aspects of mission planning for groups of unmanned vehicles, including task allocation, route planning, data collection requirements, communications planning, dynamic replanning, and multi-vehicle coordinated and cooperative planning.

Allocation determines which vehicle will perform which mission tasks. Route planning determines the path the vehicle will follow and may need to take into account factors such as the vehicle's tasks, terrain, restricted areas and no-fly zones, vehicle performance, environmental factors such as weather or ocean currents, multi-spectral signature information, threats, and payload capabilities/imaging quality requirements. Data collection planning includes sensor control and managing imaging requirements. Communications planning deals with how and when to transmit data and takes into account issues such as potential line of sight link locations, satellite availability, and communications frequencies.

Dynamic replanning involves replanning the mission after vehicles are underway. Replanning may be triggered by a wide range of factors, including new threats or targets, changes in no-fly zones or rules of engagement, new mission tasks, new intelligence or BDA data, change in the health and status of a vehicle, or loss of communications. The time frame to react to changes will dictate the type of replanning that is possible. For example, if a vehicle must react in seconds to avoid a threat, then an evasive maneuver may have to be executed, possibly followed by replanning the vehicle's mission. If the time frame is longer, the first step in the replanning process is to analyze the change in mission quality and effectiveness because of the change in planning data. For example, if a new threat is detected but has little impact on route quality, then it may not be necessary to replan. Once the new planning data is analyzed for the impact on the current plan, replanning can be performed as needed.

Multi-vehicle coordinated and cooperative planning enables teams of UAVs to avoid conflicts and to accomplish missions that require teamwork. Task allocation must take into account cooperative behavior required to accomplish a task, such as multiple sensor looks required to identify a vehicle. Coordinating route planning includes assigning ingress/egress paths to vehicles, making sure that vehicles maintain safe distances from each other, and invoking other measures to deconflict routes, such as designating areas of operation for each vehicle or assigning set altitudes to each UAV.

2.0 The ALOA SBIR Effort

2.1 Supporting Human Factors Experimentation

Several efforts are underway to design control stations for multi-Unmanned Air Vehicle (UAV) supervisory control, with the goal of increasing the operator *span of control* – the number of aircraft controlled by an operator. One example is the Operator/Vehicle Interface (OVI) program being conducted at AFRL. There are human factors issues to work out in the design of the multi-UAV control stations, including the types of tools and the content and format of information provided to the operator. Increasing the level of autonomy for operator tasks has the potential to increase span of control, but higher autonomy levels have implications for operator situation awareness, workload, and decision making. It is well documented that operators who act as passive monitors of automated systems often suffer from the "Out-of-the-Loop" (OOTL) performance problem, in which the human monitor is slow to detect problems with the system and to assume control of the system. This is a well-documented problem with regard to supervisory control. Adaptive autonomy levels have the potential to minimize the OOTL problem; however, more research is needed to determine the optimal coupling of human input and autonomous operations.

Human factors researchers need a test bed to explore issues related to span of control and autonomy. The test bed must allow researchers to present subjects with different mission planning scenarios; script events to occur during the experiment, such as pop-up threats that require the operator to replan vehicle routes; perform tests of operator situation awareness during an experiment; and record all experiment events, including operator actions and the results of those actions.

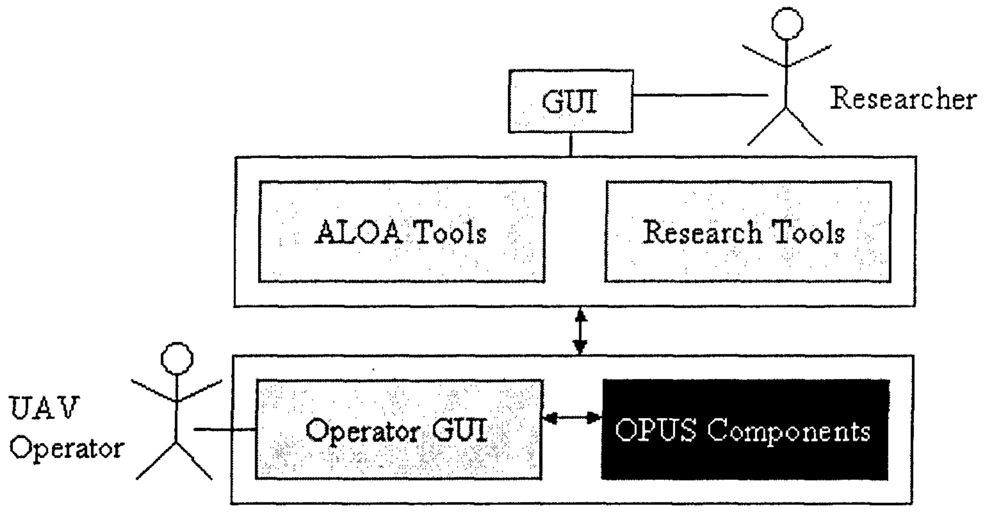
The focus of this effort is not to design a multi-UAV control station, but rather to provide an architecture and test bed for researchers to experiment with autonomy concepts and human factors issues relevant to multi-UAV control issues using state-of-the-art mission planning tools. The lessons learned through experimentation using these tools will help designers build the UAV control stations of the future.

2.2 ALOA Overview

The goal of the SBIR effort is to develop an architecture for implementing and evaluating a range of adaptive levels of autonomy for UAV supervisory control and a Multi-UAV Control Station Emulator (MCSE) test bed to demonstrate the architecture. The tools produced in this effort will provide an environment to test autonomous control strategies and the role of the human in optimizing system performance. Mission-phase based and situation-specific adaptive autonomy has the potential to keep the operator's workload manageable and to maintain the operator's situation awareness, two key aspects of effective supervisory control. The MCSE test bed will also provide an environment to test mission planning components, as well as situation awareness tools that will help increase the operator's span of control.

ORCA will employ a spiral process of design, testing, and refinement that will use input from AFRL/HECI to help drive the development process. The results of this effort will be a mature architecture for adaptive Levels of Autonomy (LOA) and a fully functioning research test bed. To illustrate the architecture, a high-level diagram is provided below. (A more detailed design is given in Section 3.) The ALOA tools will implement the architecture for adaptive levels of autonomy. The researcher will be provided tools to set up experiments and tests, and log data from experiments. The UAV operator will have access to decision aids, visualization tools, planning data, and mission planning tools through the Operator GUI.

ORCA Planning and Utility System (OPUS) components will be utilized for mission planning, data management, and simulation. OPUS is certified for operational use by the Air Force and endorsed by the Navy's UAV Advanced Technology Review Board. OPUS uses vehicle-threat interaction models that take into account vehicle Radar Cross Section (RCS) data and radar Vertical Coverage Diagram (VCD) data. OPUS state-of-the-art tools and models will provide the MCSE test bed with a realistic operational mission planning environment, which is needed for meaningful experiments and research.



One of the challenges of this effort is defining and implementing LOA. For route planning, the OPUS autorouter allows fully autonomous route planning. Experience with operators through the J-UCAS effort and other programs gives us first-hand knowledge of the types of tools and decision aids operators need when building and assessing mission plans, which supports manual mission planning. These experiences and software tools allow us to provide the capability to implement the two autonomy extremes, manual and fully autonomous, for route replanning. OPUS can also provide autonomous allocation planning. Implementing intermediate LOA requires using characteristics of both manual and autonomous task execution. Having the two extremes covered enables us to support intermediate LOA.

2.3 Phase I Technical Objectives

Below is the list of the technical objectives for Phase I:

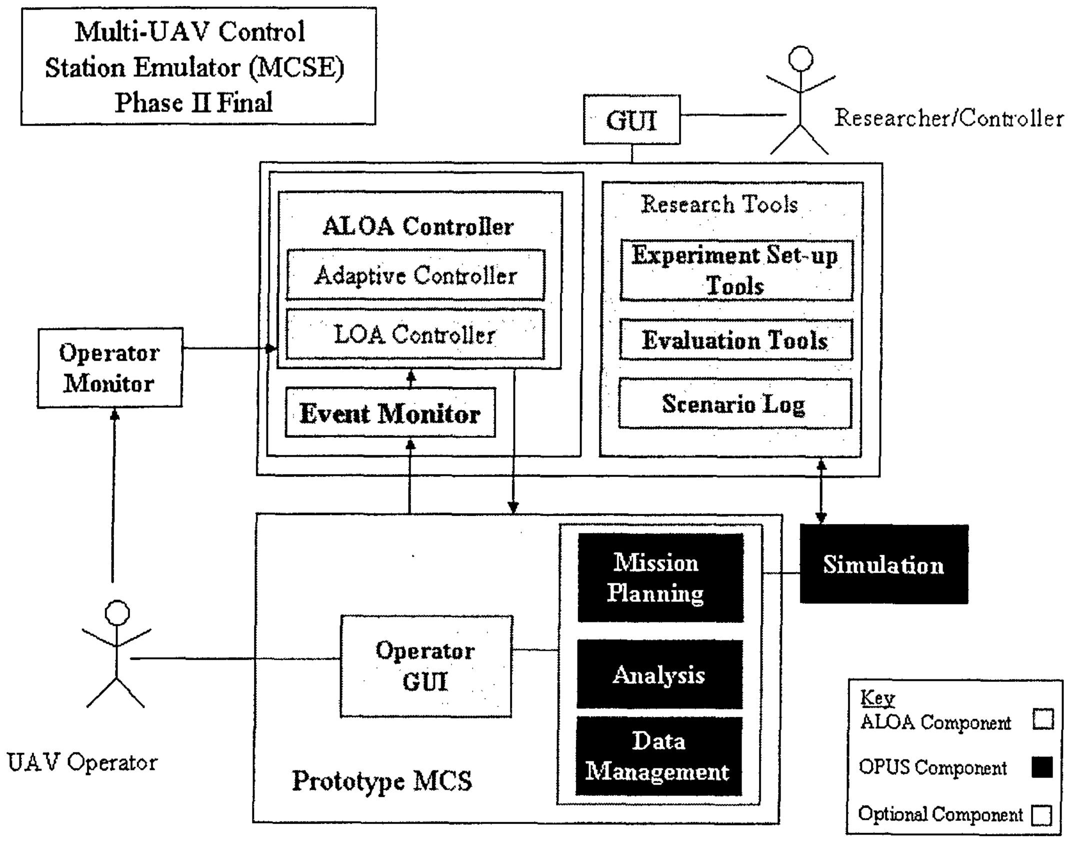
1. Devise system architecture to implement and manipulate adaptive levels of autonomy
2. Validate the architecture with a proof of concept demonstration
3. Deliver a prototype Multi-vehicle UAV Control Station Emulator (MCSE) research test bed
4. Demonstrate architecture using the prototype MCSE research test bed

We describe how we achieved these technical objectives in the sections that follow.

3.0 ALOA Architecture and MCSE Test Bed Design

One of the most important outputs of the Phase I effort was the design of the ALOA architecture and MCSE test Bed software, described in this section.

The diagram below shows high-level design for the MCSE test bed. The MCSE test bed software is being designed to run on a single machine. Two players interact with the MCSE: a researcher uses a Graphical User Interface (GUI) to set up experiments and tests and collect data; the UAV operator uses mission planning tools to control a group of aircraft in the experiment scenario.



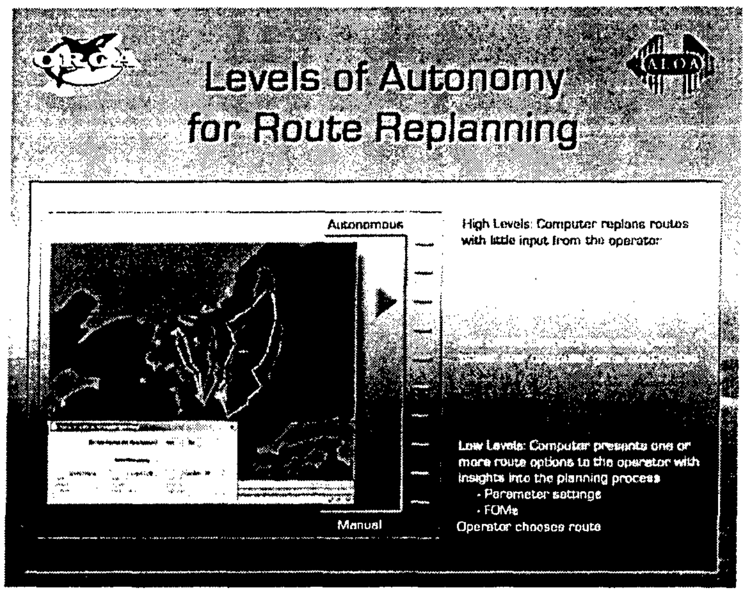
The MCSE will include components specifically designed for this effort and ORCA-developed technologies for mission planning, analysis, and simulation. Gray boxes indicate components of the system to be designed during this effort. These components will be delivered to AFRL/HECI. Black boxes represent existing ORCA-proprietary OPUS software components. The diagonally striped Operator Monitor box represents an optional component that could be developed by a third party. The components of the MCSE are described in more detail below.

3.1 ALOA Tools

The ALOA architecture consists of Levels of Autonomy (LOA) and the software components that implement the LOA. The ALOA Controller is the software component that controls the implementation and assignment of the LOA and enables adaptive LOA. The Event Monitor assists the ALOA Controller by monitoring scenario events and operator tasks. The components of the ALOA architecture are described in more detail below.

3.1.1 Levels of Autonomy (LOA)

In Phase I, we defined multiple LOA for four operator tasks: allocation, route planning, weapon control, and imagery analysis. The initial version, presented in the Technical Report delivered in Phase I, defined ten levels of autonomy for each of the four operator tasks based on Sheridan's model. We are in the process of refining our LOA models based on further research and feedback from AFRL/HECI.


Autonomous
Manual

Levels of Autonomy for Route Replanning

High Levels: Computer replans routes with little input from the operator

Low Levels: Computer presents one or more route options to the operator with insights into the planning process
- Parameter settings
- FOMs

Operator chooses route

In section 5, we give a complete list of LOA for each of the four operator tasks. This set of LOA spans the range from manual to autonomous and provides a starting point for experimentation in Phase II. Experimentation and evaluation throughout Phase II will help us refine the number of LOA and their definitions.

3.1.2 ALOA Controller

LOA can vary by task and by sortie. Route planning for one sortie could be manual, while for another sortie it could be fully autonomous. Weapon control and imagery analysis LOA could vary for the same sortie. For example, if a sortie has a weapon release against a low value target with little chance of collateral damage, that task could have a high level of autonomy, while a weapon release against a high value target may be controlled manually. The ALOA Controller, which consists of the LOA Controller and the Adaptive Controller, provides the framework for implementing adaptive LOA.

The LOA controller implements the LOA definitions. The LOA Controller determines what information is displayed to the operator and what tasks the automated mission planning tools perform for a given autonomy level. It also contains decision logic to select plans for tasks with high LOA.

The Adaptive Controller receives inputs from the Event Monitor about new events and changes in mission phase and will use this information to adapt the LOA. Current tasks performed by the operator must be factored into any decision about adapting LOA.

3.1.3 Event Monitor

The Event Monitor monitors the scenario for changes in mission phase (ingress, egress, target area, etc.) and events, such as popup threats, new mission tasks, or loss of a vehicle, and feeds this information to the ALOA Controller.

3.1.4 Operator Monitor

We reserve a place in the architecture for a component that monitors biometric data to measure the operator's workload and performance. This component would feed information to the ALOA controller. ORCA will not develop this component, but we will provide an interface to the ALOA Controller if a third party were to provide such a tool.

3.2 Mission Planning Tools

The ORCA Planning and Utility System (OPUS) includes state-of-the-art mission planning and analysis tools. The operator will utilize OPUS tools to allocate tasks to vehicles, develop route plans, evaluate vehicle-threat interactions, simulate flyouts of mission plans against the threat laydown, and analyze mission plans. The MCSE test bed uses OPUS data management tools to manage mission planning data for experiments. Below is a brief description of OPUS components and functionality.

3.2.1 Allocation

The OPUS target allocation component assigns vehicles to sets of mission objectives to achieve force application goals. The assignment process considers vehicle resources such as fuel and available weapons and sensors, assignment costs such as vehicle value and probability of arrival, and target values. During the allocation process, threat and vehicle models have been adapted to provide a "good enough" route planning answer, leaving details to the route planner.

3.2.2 Autorouting

Route planning is the lowest level of the planning process. Given an ordered set of tasks or objectives, the autorouter considers vehicle performance, threat susceptibility, and terrain to develop a feasible, survivable route that achieves the mission objectives. Mission objectives include weapon releases, pre-strike or post-strike bomb damage assessment, sensor imaging assignments, and surveillance and reconnaissance tasks. ORCA's autorouting algorithms are fast enough to generate new routes in response to changes in the threat environment or mission objectives.

3.2.3 Analysis

OPUS includes a suite of analysis tools that will be available to the operator. OPUS provides analysis tools and decision aids that highlight information not ordinarily available or visible.

3.2.3.1 Mission Effectiveness Metric

The OPUS autorouter generates routes that attempt to achieve mission objectives while also maximizing survivability. OPUS uses the mission planning Figure of Merit (FOM) Expected Weapons on Target (EWOT) as the objective function for route planning to capture these goals. EWOT is defined by

$$EWOT = \sum_{j=1}^{n} w_j \prod_{i=1}^{j} P_{s_i}$$

where w_j is the weight, or value, mission objective j, and P_{s_i} is the probability of survival for the ith segment of the route.

3.2.3.2 Attrition Metrics

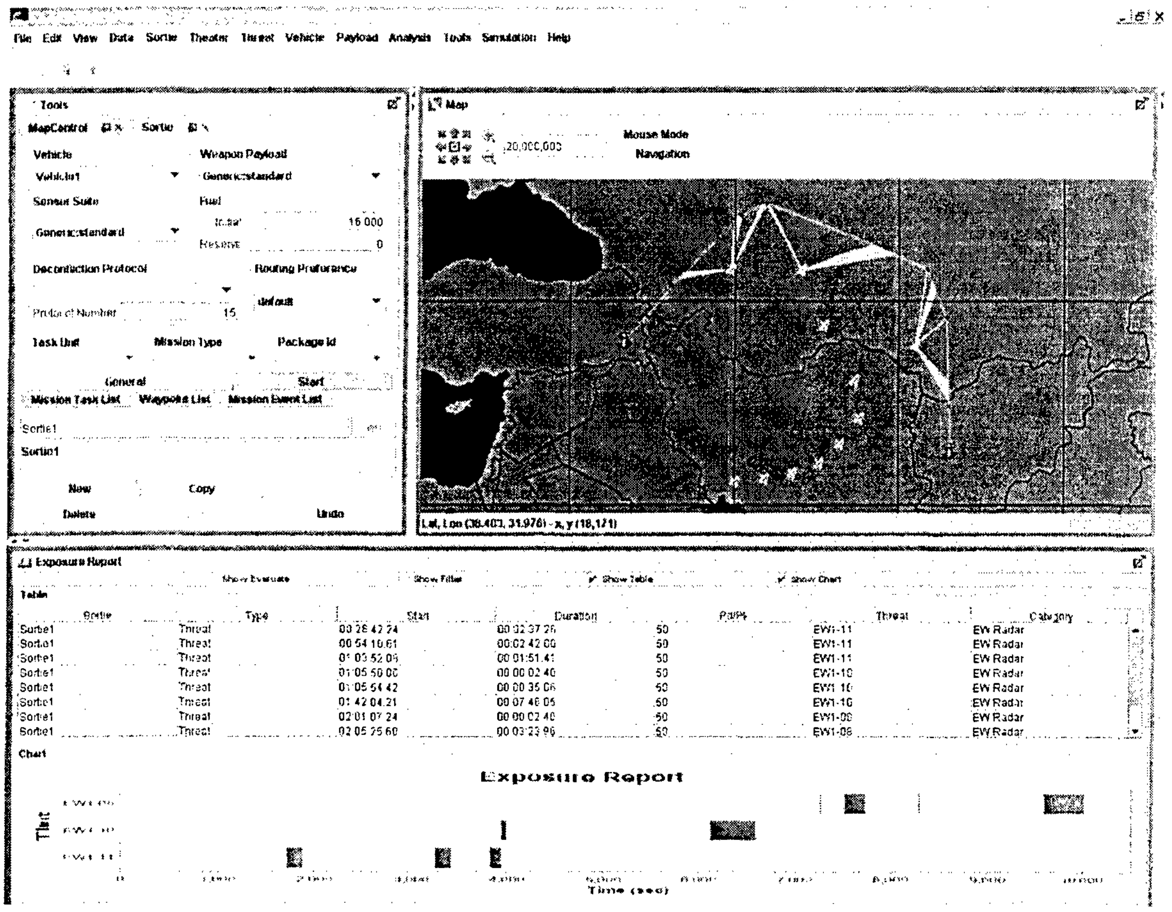
The probability of survival for a sortie route is the primary attrition metric. Additional route FOMs are provided including the number of SAM shots, amount of exposure to EW/GCI radars, amount of exposure to tracking radar, and track time by sites and networks.

3.2.3.3 Route Traversal

The traversal function in this tool makes the aircraft's radar threat template visible on the computer screen. Any radar on the ground that is highlighted by that template will have the potential to spot the vehicle. Other templates can be used to show where there is the threat of a missile launch, or where the vehicle will be able to take required surveillance images, or effectively release weapons. Threat templates, weapon footprints, and sensor envelopes can be displayed so the operator can investigate threat interactions and weapon and sensor plans.

3.2.3.4 Threat Evaluation

OPUS evaluation tools provide information about how a route interacts with threats, such as exposures to EW and SAM tracking radar and SAM sites, for multiple levels of probability of detection and probability of kill. Reports on exposure of sortie routes to any combination of threat types can be generated. The dialog below shows a map display with a graphical display of exposure to threats using "threat fans", which gives the operator a quick sense of how well threats can see the vehicle. If a route is replanned, either manually by dragging the route or using the autorouter, the exposure reports will be updated, giving the operator immediate feedback about route quality.



Vehicle: Vehicle1
Weapon Payload: Generic:standard
Sensor Suite: Generic:standard
Fuel: Initial 16 000; Reserve 0
Deconfliction Protocol
Routing Preference: default
Protocol Number: 15
Task Unit
Mission Type
Package Id
General
Start
Mission Task List
Waypoint List
Mission Event List
Sortie1
Sortie1
New
Copy
Delete
Undo

Lat, Lon (38.401, 31.976) - x, y (18,171)

Table

Sortie	Type	Start	Duration	Pd/Pk	Threat	Category
Sortie1	Threat	00 28 42 74	00 02 37 26	50	EW1-11	EW Radar
Sortie1	Threat	00 54 10.61	00:02 42 00	50	EW1-11	EW Radar
Sortie1	Threat	01 03 52 09	00 01:51.41	50	EW1-11	EW Radar
Sortie1	Threat	01 05 50 00	00 00 02 40	50	EW1-10	EW Radar
Sortie1	Threat	01 05 54 42	00 00 35 06	50	EW1-10	EW Radar
Sortie1	Threat	01 42 04.21	00 07 46 05	50	EW1-10	EW Radar
Sortie1	Threat	02 01 07 24	00 00 02 40	50	EW1-09	EW Radar
Sortie1	Threat	02 05 25 60	00 03 23 96	50	EW1-08	EW Radar

3.2.4 Data Management

The OPUS data management component manages the data required for the OPUS mission planning and analysis components. ORCA will provide data and sample cases for experiments.

3.2.5 Simulation

Experiment scenarios can be played out using OPUS simulation tools. The simulation component implements a discrete event simulation to model the execution of sorties against the threat laydown.

3.2.6 UAV Operator Graphical User Interface (GUI)

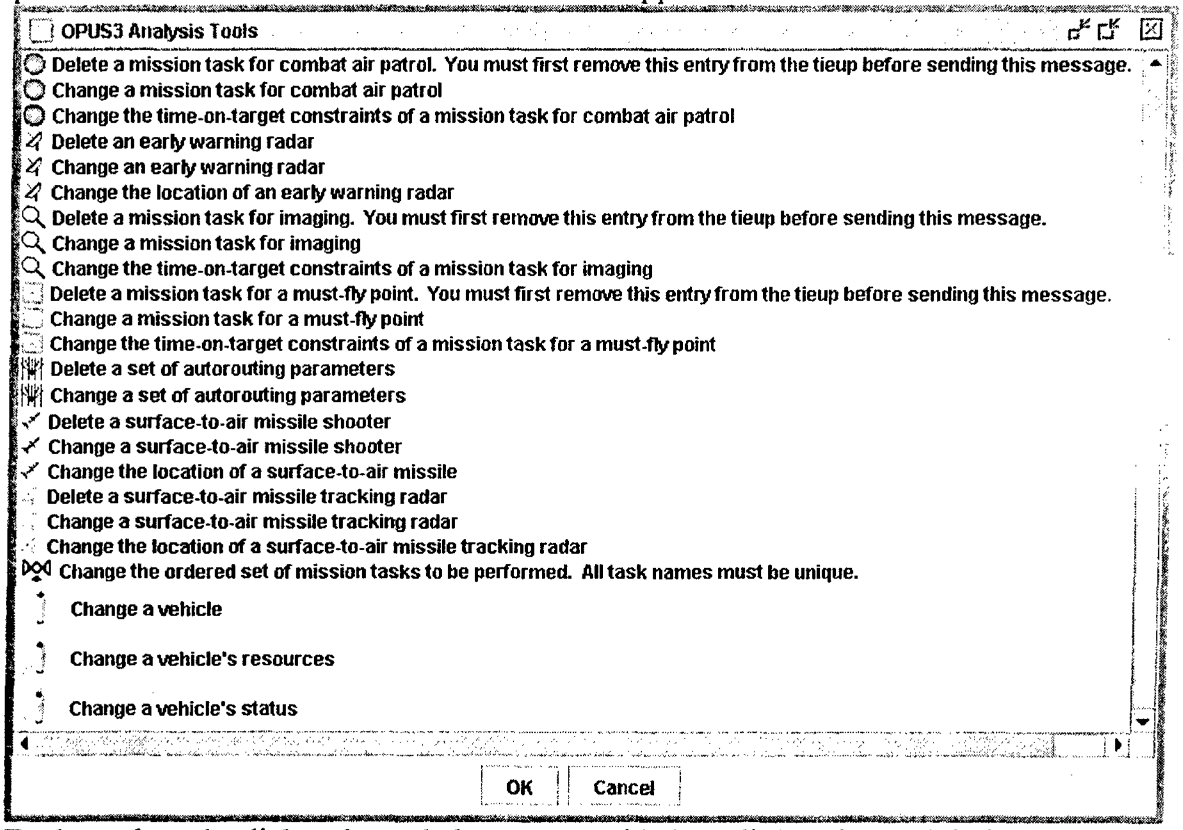
The operator's Graphical User Interface (GUI) includes decision aids and tools to access OPUS mission planning and analysis components. The GUI will be designed for a dual screen display. To provide experimental flexibility, the GUI will have multiple configurations.

3.3 Tools for the Researcher

3.3.1 Experiment Setup Tools

The Scenario Generator and Script Editor are the tools used to set up experiments. The Scenario Generator is used to define all mission planning data for an experiment, including threats, targets, the number and type of vehicles assigned to the operator, mission tasks, such as weapon releases and sensor imaging assignments, and geographical data such as terrain and no-fly zones. The scenario also specifies the operator GUI configuration and quality level settings for automated mission planning tools. The script editor is used to script events to occur during the experiment, including mission events such as popup threats, new targets, and loss of a vehicle. The script specifies the times and locations for the events. The script also specifies the human factors tests to administer during the experiment. The experiment setup tools streamline the process of building scenarios and adding events.

ORCA has built scenario generation and script editing tools for other efforts. The dialog below shows a menu used in one of our scenario generation tools to select events for a mission scenario. A similar menu of choices for building scenarios is included in the script editor. Using such a tool, the user can specify the threat type and specify parameters for when and where the threats will appear.

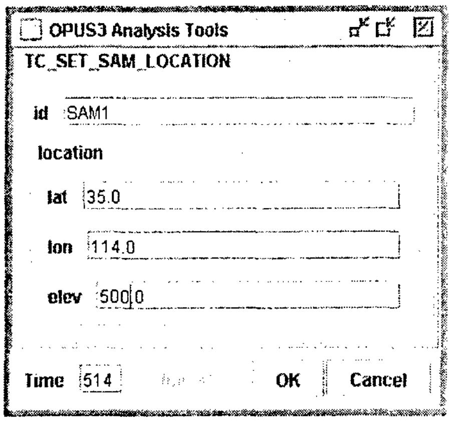


OPUS3 Analysis Tools

- Delete a mission task for combat air patrol. You must first remove this entry from the tieup before sending this message.
- Change a mission task for combat air patrol
- Change the time-on-target constraints of a mission task for combat air patrol
- Delete an early warning radar
- Change an early warning radar
- Change the location of an early warning radar
- Delete a mission task for imaging. You must first remove this entry from the tieup before sending this message.
- Change a mission task for imaging
- Change the time-on-target constraints of a mission task for imaging
- Delete a mission task for a must-fly point. You must first remove this entry from the tieup before sending this message.
- Change a mission task for a must-fly point
- Change the time-on-target constraints of a mission task for a must-fly point
- Delete a set of autorouting parameters
- Change a set of autorouting parameters
- Delete a surface-to-air missile shooter
- Change a surface-to-air missile shooter
- Change the location of a surface-to-air missile
- Delete a surface-to-air missile tracking radar
- Change a surface-to-air missile tracking radar
- Change the location of a surface-to-air missile tracking radar
- Change the ordered set of mission tasks to be performed. All task names must be unique.
- Change a vehicle
- Change a vehicle's resources
- Change a vehicle's status

OK Cancel

Tools, such as the dialog shown below, are provided to edit location and timing information for the events.


OPUS3 Analysis Tools

TC_SET_SAM_LOCATION

id SAM1

location

lat 35.0

lon 114.0

elev 500.0

Time 514

ORCA will implement a scenario rehearsal capability allowing the researcher to review scripted events in the operator GUI before performing an experiment.

3.3.2 Evaluation Tools

Several tools are included in the MCSE test bed to allow human factors experts to perform tests and measure operator performance. The experimenter specifies tests to be used in an experiment and any test parameters in the scenario script. Results of the tests are saved in a file. In the Phase I prototype software, ORCA implemented the functionality to perform Situation Awareness Global Assessment Technique (SAGAT) tests and secondary task tests, and allow the researcher to pause an experiment to perform other tests.

For the SAGAT test, the operator's screen is blanked, a set of questions provided by the researcher is displayed, and the operator enters answers on the screen using the keyboard. The operator's answers will be saved in a file.

Two secondary task tests will be included in the MCSE test bed: an "unidentified vehicle" test and a "Mission Mode Indicator" (MMI) test. For the unidentified vehicle test, an aircraft, tank, jeep, or other vehicle will be displayed on the screen. The operator must click on the vehicle to make it disappear. The time from the appearance of the vehicle until the operator clicks it will be recorded. The parameters for this test are the start and end time, and the vehicle type to display. The start time and length of the test can also be selected at random. The unidentified vehicle test has been implemented in the Phase I prototype software.

For the Mission Mode Indicator (MMI) test, a series of lights (green, yellow, and red) are displayed in the GUI. The green light is lit initially. As the test runs, the color changes to yellow and then to red, after preset time delays. When the operator recognizes that the color is no longer green the requirement is to click on any light, after which a dialog pops up and displays a randomly generated number. The operator must type the number into the field provided in the dialog. The length of time from the appearance of the dialog until the operator enters the number will be recorded, and the number entered. The parameters for this test are the start and end times, the time for yellow light display and the time for the red light display. This test can also be run randomly.

In addition to these testing capabilities built into the software, the researcher can pause the experiment to administer other types of tests. When the experiment is paused, the simulation is paused and the operator's screen goes blank.

3.3.3 Scenario Log

The scenario log provides a data logging capability for experiments. Data recorded includes scenario events, plans generated by the operator with Figures of Merit (FOMs), results of tests, and all operator actions and response times. ORCA will implement performance measurement tools to record mouse clicks and other operator actions, measure the time to complete tasks, and record outcomes of the operator's actions. Some of this has been implemented in the Phase I prototype software.

The scenario log will include filtering tools to allow the researcher to specify a subset of the recorded data to output at the end of an experiment. All data recorded in the scenario log can be saved in a file so that other data can be retrieved after an experiment.

3.3.4 Researcher GUI

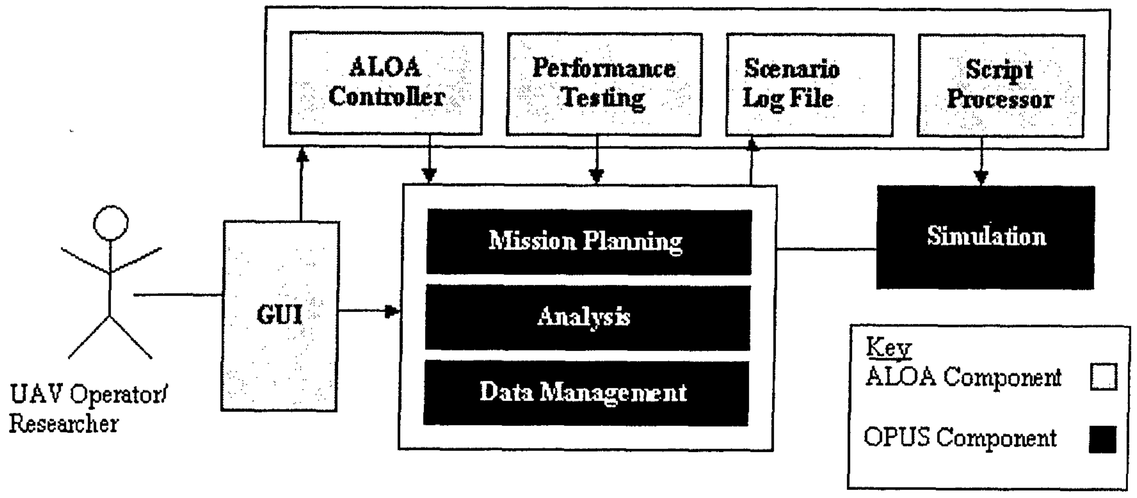
The GUI allows the researcher to access experiment setup tools, evaluation tools, and the scenario log. The GUI will have an Operator GUI view, which will be used by the researcher to view a rehearsal of an experiment scenario and script.

4.0 Prototype ALOA Software

In Phase I, ORCA designed a prototype MCSE test bed, which has a subset of the MCSE test bed components, tools, and functionality described above. The prototype software provides a limited testing environment and will serve as the basis for initial Phase II testing and experimentation.

4.1 Prototype Design

The diagram below shows the design for the Phase I prototype MCSE test bed.



Phase I Prototype Multi-UAV Control Station Emulator (MCSE) Test Bed

The ALOA Controller implements the LOA defined in Phase I. In the prototype MCSE test bed, we implemented LOA for route planning, weapon control, and imagery analysis.

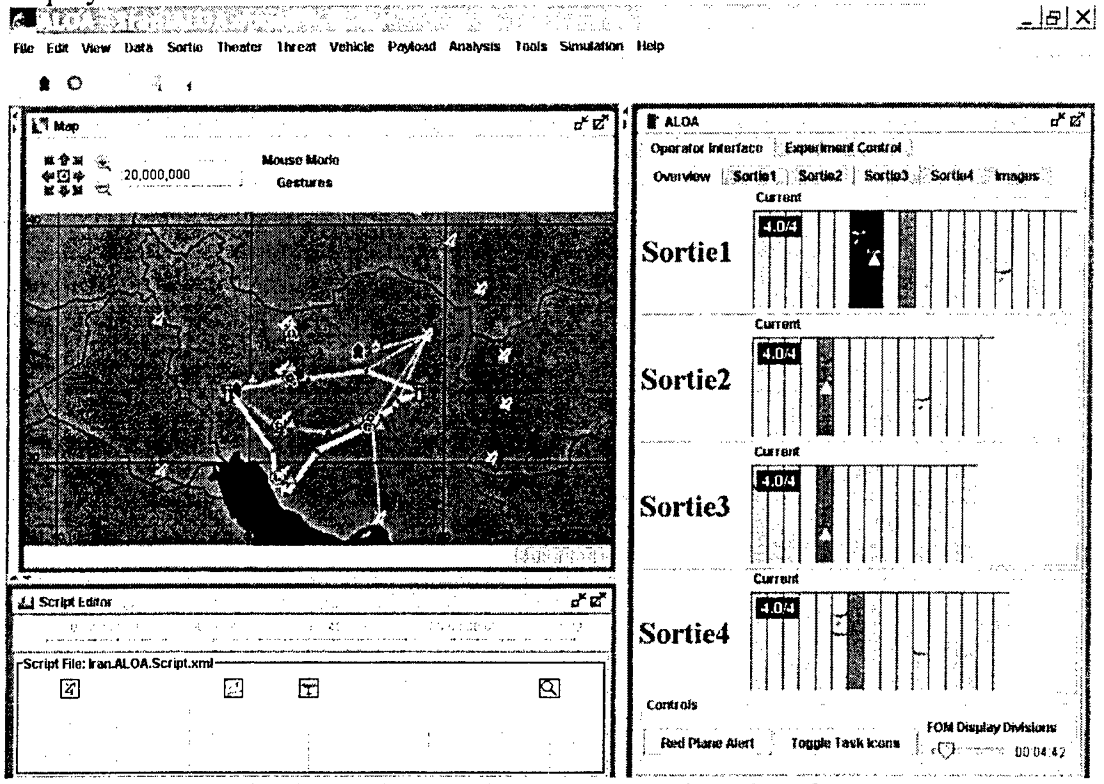
LOA are set manually, and the ALOA Controller determines tasks to be performed by the operator and the computer for a given LOA. The Evaluation Tools component implements the SAGAT capability and secondary task tests. It also allows the experimenter to pause an experiment to administer a test. The Scenario Log file includes mission planning data and FOMs and scenario events. The Script Editor allows the experiment to script events for the scenario. A single GUI incorporates both researcher and operator tools.

4.2 *Prototype Software Description*

The ALOA software prototype consists of an operator interface, a scenario and script editor, a statistics analyzer, and several scenarios and scripts. Each of these features is described in detail in the user guides included in the appendix to this document. Below we highlight some of the important features of the prototype software.

4.2.1 ALOA Graphical User Interface (GUI)

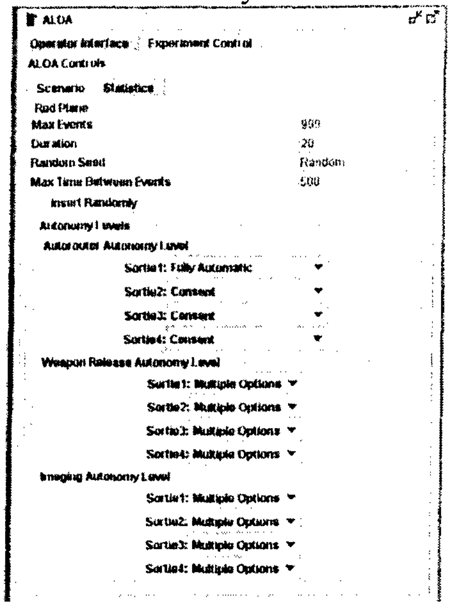
The ALOA GUI, shown below, consists of the operator interface, script editor, and map display.



The right side of this display is the primary operator interface for ALOA. The operator interface consists of a number of tabs but currently shows an overview. The additional tabs provide access to each sortie as well as to any images that are available.
The bottom of the display shows a script that executes when the simulation starts. Note that the script can be hidden so the operator cannot see events that are yet to occur. Instructions for using the script editor are contained in a companion document.
The map shows sortie routes as well as targets, threats, and airbases.

4.2.2 Experiment Control Tools

In addition to a script editor, the ALOA software prototype provides some graphical control of scenario generation. Many aspects of the scenario can be controlled via OPUS data dialogs. For instance, the creation of imaging and weapon release tasks as well as vehicle performance and other data can be controlled via the OPUS data dialogs. Other ALOA experiment controls can only currently be controlled from text files. The following shows the controls that can currently be accessed:



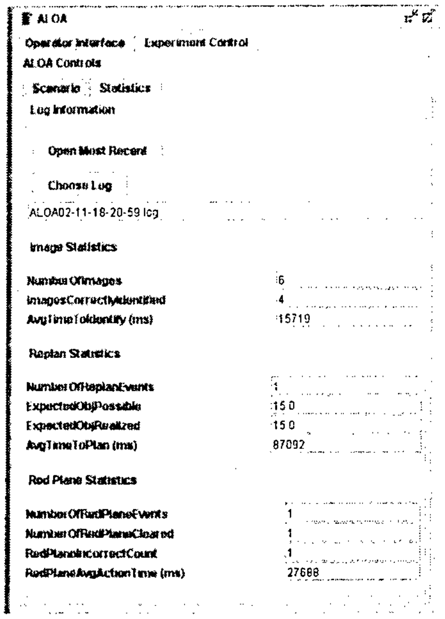
Red Plane
Max Events 999
Duration 20
Random Seed Random
Max Time Between Events 500
Insert Randomly
Autonomy Levels
Autorouter Autonomy Level
Sortie1: Fully Automatic
Sortie2: Consent
Sortie3: Consent
Sortie4: Consent
Weapon Release Autonomy Level
Sortie1: Multiple Options
Sortie2: Multiple Options
Sortie3: Multiple Options
Sortie4: Multiple Options
Imaging Autonomy Level
Sortie1: Multiple Options
Sortie2: Multiple Options
Sortie3: Multiple Options
Sortie4: Multiple Options

4.2.3 Log File

ALOA produces a log file of major events that are recorded with time stamps at the millisecond level. Events that are recorded include:

- button presses
- tab switching
- popup threats
- red plane events, and
- image identifications

The statistics are accessible through a text-based log file. The software prototype also provides a panel to access and display statistics from an arbitrary log file. The following shows statistics from a selected log file:



ALOA

Operator Interface | Experiment Control

ALOA Controls

Scenario | Statistics

Log Information

Open Most Recent

Choose Log

ALOA02-11-18-20-59.log

Image Statistics

NumberOfImages	6
ImagesCorrectlyIdentified	4
AvgTimeToIdentify (ms)	15719

Replan Statistics

NumberOfReplanEvents	1
ExpectedObjPossible	15.0
ExpectedObjRealized	15.0
AvgTimeToPlan (ms)	87092

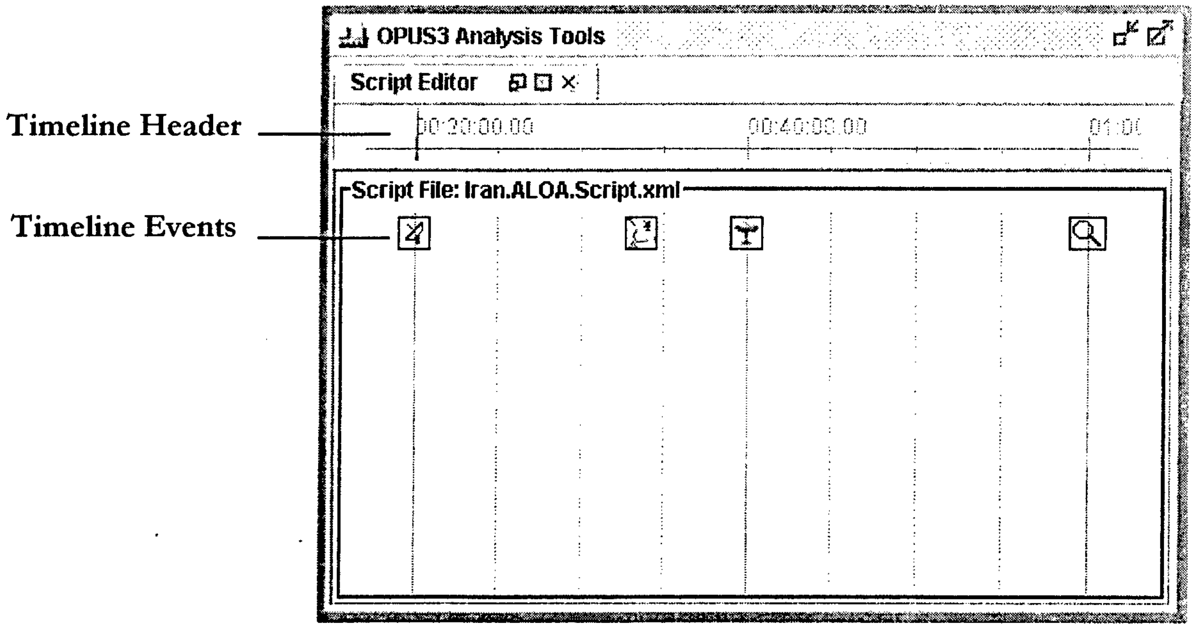
Red Plane Statistics

NumberOfRedPlaneEvents	1
NumberOfRedPlaneCleared	1
RedPlaneIncorrectCount	1
RedPlaneAvgActionTime (ms)	27688

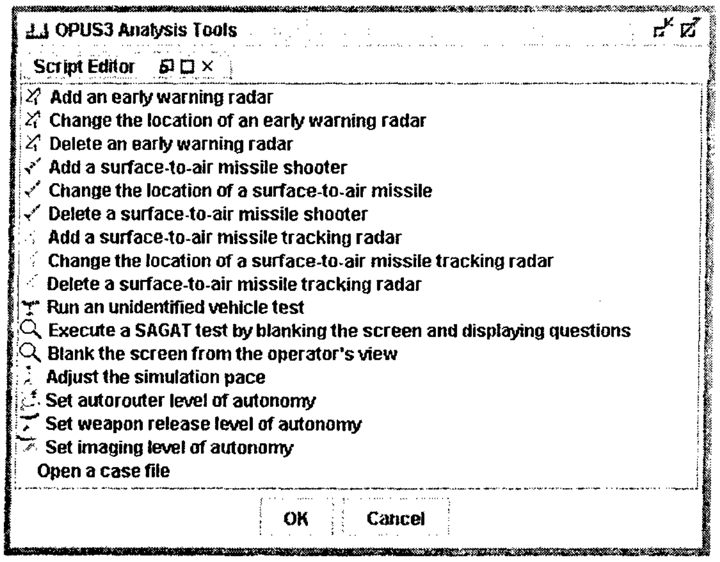
From this panel, you can access either the most recent log, which would correspond to the last run of an experiment, or you can select an arbitrary log. When a log is selected statistics are displayed. ALOA currently displays statistics regarding image identifications, replanning, and red planes.

4.2.4 The Script Editor

The script editor allows an experimenter to script events during ALOA simulation. This facilitates experimental setup and scenarios. The diagram below shows a script example.



The **Timeline Header** indicates the current simulation time and the time interval that is displayed. The **Timeline Events** section contains script events. Events can be added to the script. The menu below, accessible through the script editor dialog, lists the script events that can be inserted.



Add an early warning radar
Change the location of an early warning radar
Delete an early warning radar
Add a surface-to-air missile shooter
Change the location of a surface-to-air missile
Delete a surface-to-air missile shooter
Add a surface-to-air missile tracking radar
Change the location of a surface-to-air missile tracking radar
Delete a surface-to-air missile tracking radar
Run an unidentified vehicle test
Execute a SAGAT test by blanking the screen and displaying questions
Blank the screen from the operator's view
Adjust the simulation pace
Set autorouter level of autonomy
Set weapon release level of autonomy
Set imaging level of autonomy
Open a case file

Scripts can be saved, edited, and deleted.

5.0 Levels of Autonomy

One of the important accomplishments of Phase I was defining multiple LOA for allocation, route planning, weapon control, and sensor management. A level of autonomy defines the distribution of workload between the operator and the computer. The lowest is *manual* and the highest is *fully autonomous.* In the descriptions below, the italicized items are generic descriptions of the level of autonomy, based on definitions given by Parasuraman[3], and the bulleted items describe the level of autonomy in the context of the operator task.

5.1 Route Planning

Route planning involves generating routes that accomplish mission tasks, such as weapon releases and sensor imaging tasks. An input to the route planning process is an ordered list of tasks for each sortie, which comes from the allocation process, the ATO, or other external source.

1. Manual: operator manual planning with automated decision aiding

- Operator monitors system: data about changes (popup threats, new objectives, etc.) is available, but the system does not inform the operator explicitly
- Operator initiates planning/replanning
- Operator plans one or more routes by selecting waypoints manually, either on the map display or by entering the data by hand; OPUS aids in generating the route by computing turns and rescheduling mission tasks
- Computer evaluates the route(s) and displays FOMs for the new route(s) and the current route

[3] Parasuraman, R., Sheridan, T.B., Wickens, C.D., "A Model for Types and Levels of Human Interaction with Automation", IEEE Transactions on Systems, Man, and Cybernetics--Part A: Systems and Humans, Vol. 30, No. 3, May 2000

- Operator can use the traversal capability of the analysis component to view a flyout of each route plan
- For each sortie, the operator selects one of the new routes or keeps the current route (if replanning)

2. *Multiple options: Autorouter generates a set of route plans for each sortie*
   - Operator initiates replanning when mission data changes
   - Operator selects sets of route parameters to be used by the autorouter to generate routes
   - Autorouter generates a route for each parameter set
   - Operator can generate routes manually as well
   - Computer evaluates the routes and displays FOMs selected by the operator for each new route and the current route
   - Operator can use the traversal capability to view a flyout of each route plan
   - Operator can modify the routes manually, try other parameter sets and use the autorouter to generate new routes
   - Operator selects one of the new routes or keeps the current route

3. *Autorouter suggests one alternative*
   - Computer initiates replanning when mission data changes
   - Autorouter generates a new route
   - Computer evaluates the route and displays FOMs for the new route and the current route
   - Operator can use the traversal capability of the computer to view a flyout of the route plans
   - Operator accepts or rejects new route
4. *Operator veto: computer executes that suggestion if the operator approves*
   - Computer initiates replanning when mission data changes
   - Autorouter generates a new route
   - Computer evaluates the route and displays FOMs for the new route and the current route
   - Computer chooses between the new route and the current route
   - Operator can veto the plan
5. *Timed veto: allows the operator a restricted time to veto before automatic execution*
   - Computer initiates replanning when mission data changes
   - Autorouter generates a new route
   - Computer evaluates the route and displays FOMs for the new route and the current route
   - Computer accepts new route or keeps current route
   - Timed consent mode: operator can veto new plan within *t* seconds/minutes after replanning is complete; if the operator takes no action in the decision window, the plan is executed autonomously
6. *Fully autonomous*
   - Computer initiates replanning when mission data changes

- Computer generates a new route
- Computer evaluates the new route
- Computer accepts new route or keeps current route

5.2 Imagery Analysis

Imagery analysis involves analyzing sensor images and identifying objectives in the images. Automatic Target Recognition (ATR) technology provides automated imagery analysis capability.

1. *Manual*
    - Operator analyzes images
    - Operator controls the sensors during mission execution
    - ATR is not used
2. *Computer offers a set of decision/action alternatives*
    - Computer analyzes images and provides operator with a list of possible identifications with associated probability of correct identification
    - Operator chooses from the list of possible identifications or identifies the objects himself
3. *Suggests one alternative*
    - Computer analyzes images and provides operator with a possible identification and probability of correct identification
    - Operator identifies the object
4. *Operator veto: computer executes that suggestion if the human approves*
    - ATR identifies objects in the imagery
    - Operator must approve the ATR output before it is used for planning
5. *Computer allows the human a restricted time to veto before automatic execution*
    - ATR identifies objects in the imagery
    - Operator has *t* seconds to approve the ATR output before it is used for planning
    - Otherwise, the ATR output is accepted
6. *The computer decides everything, acts autonomously, ignores the human*
    - ATR identifies objects in the imagery

5.3 Weapon Control

Weapon control is the task of deciding if a planned weapon release should be executed. Before a weapon release, intelligence data and imagery can be reviewed to determine if the target has moved and if the weapon will strike the desired target, and whether there is potential for collateral damage; authorization for the weapon release can be manual or autonomous.

1. *Manual*
    - Operator monitors weapon release tasks
    - Operator analyzes target, weapon, and intelligence data before a weapon release
    - Operator must authorize the weapon release
2. *Operator authorization: computer executes that suggestion if the human approves*
    - Computer informs the operator before a weapon release and provides updates on weapon, target, and intelligence data
    - Operator must authorize the weapon release

3. *Operator veto: Computer allows the human a restricted time to veto before automatic execution*
   - During mission execution, the computer informs the operator before the weapon release and provides updates on weapon, target, and intelligence data
   - Operator can stop the weapon release up to *t* seconds before the planned release
   - Otherwise, weapon release is performed autonomously, according to the route plan
4. *Autonomous: The computer decides everything, acts autonomously, ignores the human*
   - The computer plans the weapon release
   - Weapon release is performed autonomously, according to the route plan

5.4 Allocation

Allocation is the process of assigning ordered sets of mission tasks to vehicles.

1. *Manual allocation with automated decision aids*
   - Operator initiates the allocation process
   - Operator allocates tasks to each sortie
   - Operator can use the route planner to generate and analyze routes based on the allocation
   - Operator can reallocate
2. *Computer offers a set of allocations*
   - Operator initiates the allocation process
   - Operator chooses predefined parameter sets or defines parameter sets for allocation
   - Computer generates allocations for each parameter set
   - Operator chooses one of the allocations generated by the computer
3. *Computer suggests an alternative*
   - Computer generates an allocation and an alternative
   - Operator chooses one of the allocations generated by the computer
4. *Operator veto: operator must approve the allocation before it is sent to the autorouter*
   - Computer initiates the allocation process
   - Computer generates an allocation
   - Operator must approve the allocation before route planning is performed
5. *Allows the human a restricted time to veto before automatic execution*
   - Computer initiates the allocation process
   - Computer generates an allocation
   - Operator is notified of the plan and is given *t* seconds to veto it
   - Otherwise, allocation is passed to the route planner
6. *Autonomous allocation*
   - System initiates the allocation process
   - Allocation component generates an allocation and sends it to the route planner

6.0 Conclusion

In Phase I, ORCA devised the ALOA architecture for testing and evaluating different methods for adaptive levels of autonomy. We defined multiple LOA for each of four operator tasks: allocation, route planning, imagery analysis, and weapon control. To demonstrate the architecture and the implementation of the LOA, we designed a prototype Multi-UAV Control Station Emulator (MCSE) research test bed, by building on existing ORCA-developed software components.

Our Phase I work has built a solid foundation for a successful Phase II. In Phase II, we will mature the ALOA architecture, including allowing contingency-specific adaptation of LOA. We will finalize the design of the MCSE test bed and existing MCSE components, and design new components to provide a fully functional experimentation and research test bed environment. At the end of Phase II, we will perform a full evaluation of the ALOA architecture in a representative high-fidelity UAV simulation environment.

Appendix

ALOA Prototype User Guide

1.0 ALOA Software Prototype

The ALOA software prototype consists of an operator interface, a scenario and script editor, a statistics analyzer, and several scenarios and scripts. This document describes:

- Starting ALOA
- Using ALOA
- Predefined scenarios and scripts
- Files Included on the CD

Section 2 of the Appendix describes the script editor, which can be used to generate scripted events in a simulation. Several scripts, however, have already been created and are provided with this software prototype.

1.1 Starting ALOA

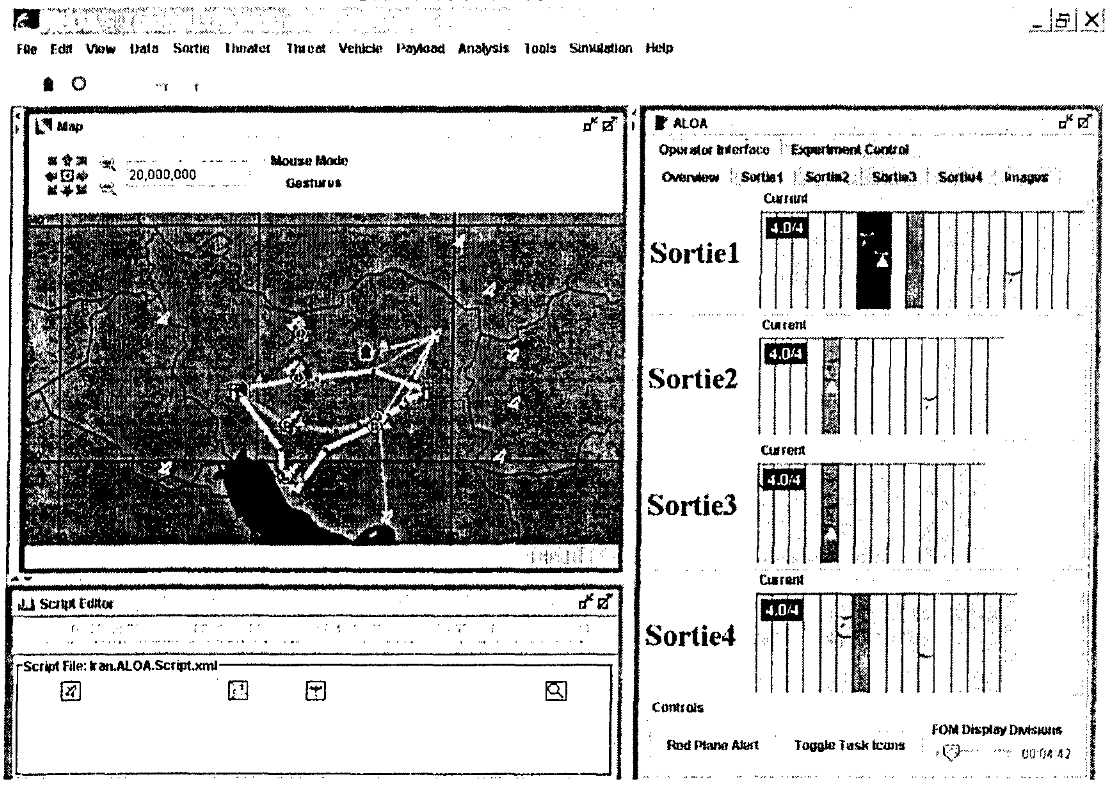
To start, run ALOA.bat in the base directory. When the program is run for the first time, a window appears asking for a license. Call ORCA at (562) 907-6700 and provide the Host ID listed in the window. Once you obtain a license, enter it into the license field, then click **OK**.

The minimum required to experiment with ALOA is to open Traversal from the Analysis menu and click the **Run** button. This starts execution of the scenario and script that ALOA has opened by default. You can then explore the various dialogs and panels during the simulation. This document provides a more detailed description of the dialogs available in ALOA.

ALOA automatically starts up with a scenario. To change this scenario, edit the DEFAULT_CASE variable in ALOA.bat. The scenarios included with ALOA are discussed later in this document.

1.2 Using ALOA

By default, ALOA starts up with the case Iran.ALOA. You can open other cases by going to the File menu and clicking **Open**. Three cases are provided with this prototype, and you can access them from that menu. Upon opening ALOA, the operator interface, script editor, and map are visible to the user. In particular, the following display is shown:



The right side of this display is the primary operator interface for ALOA. The operator interface consists of a number of tabs but currently shows an overview. The additional tabs provide access to each sortie as well as to any images that are available.
The bottom of the display shows a script that executes when the simulation starts. Note that the script can be hidden so the operator cannot see events that are yet to occur. Instructions for using the script editor are contained in a companion document.
The map currently shows all four sortie routes as well as targets, threats, and airbases that are part of the scenario.

1.2.1 Operator Interface

The operator interface consists of:

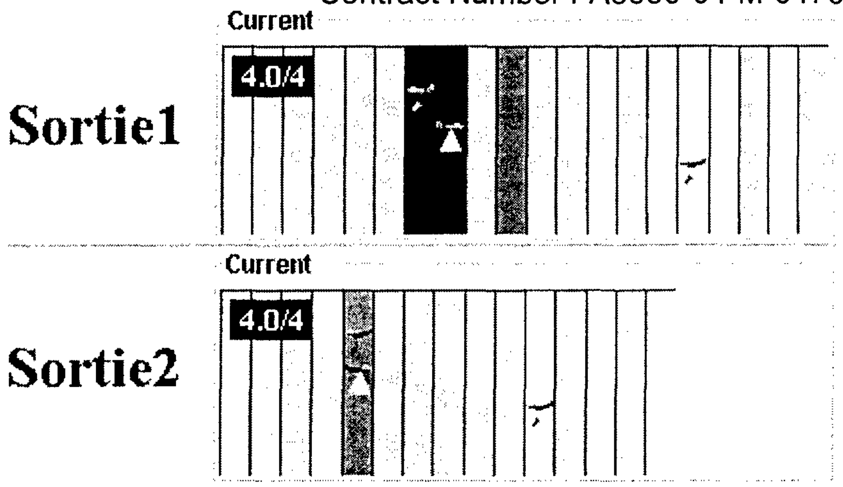
- an overview
- sortie specific tabs
- image retrieval

1.2.1.1 Overview

The overview provides a graphical summary of the quality of each sortie's route as well as an indication of when objectives will occur. The graphical summary is known as the Figure of Merit Display Panel (FDP), which is explained next.

1.2.1.2 Figure of Merit Display Panel

The Figure of Merit Display Panel (FDP) is a technique to provide an operator with insight regarding the quality of a route. Consider the following FDP:



Each rectangle in the display represents a period of time along the route. In this case, each rectangle represents about 4 minutes and 40 seconds. The rectangles are color coded to provide insight into the danger level during that time of the route. The color coding scheme is as follows:

1. Black: SAM exposure
2. Red: Tracker exposure
3. Yellow: EW exposure
4. Green: No exposure
5. White: Not flying

The colors are ordered, with the highest priority color taking precedence. For example, if there is both SAM and tracker exposure during a 4 minute and 40 second period of the route, then the rectangle is colored black. In addition, threshold values can be specified in a configuration file, which indicate how much exposure should exist before coloring a rectangle. For instance, if the EW exposure threshold is set to 20 seconds then there must be at least 20 seconds of exposure before a rectangle becomes yellow.

The entire display represents the maximum length of all routes under control. In this case, Sortie2 finishes before Sortie1, which is the reason for white space at the end of the route. Also shown on the FDP are mission task icons. These icons indicate that an image or weapon release will occur during that period of time. Note that these icons can be toggled on and off from the overview panel in the operator interface.

Notice that numbers are displayed in the top left corner of the FDP. These numbers indicate the expected number of objectives achieved during the route. The expected number of objectives achieved is the primary metric to determine a route's effectiveness. A route that is expected to achieve more objectives than another is typically the best choice. Note that finishing a route is considered an objective, which is the reason that both sorties show 4/4. Each sortie has two weapon releases, one image, and both have to finish the route.

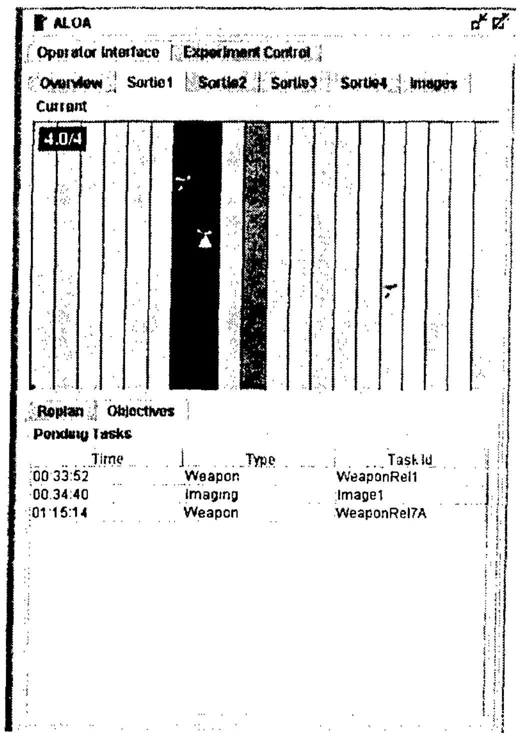
The expected objectives achieved that appear in the FDP can be programmatically degraded, however. The probability that the value is degraded can be specified in an ALOA configuration file, which is discussed later.

During the simulation, a small blue dot appears at the bottom of the FDP. The blue dot represents the current time and allows you to identify which rectangle an aircraft is currently traveling through.

The duration of each rectangle displays in the overview panel. The number of rectangles in the FDP can be adjusted as well, which provides more or less visibility into the route.

1.2.2 Sortie Panels

There is a panel for each sortie under the operator's control. Each panel consists of the FDP of the route as well as a table of pending objectives. The following shows the FDP of Sortie1's route as well as its pending objectives:



Pending Tasks

Time	Type	Task Id
00:33:52	Weapon	WeaponRel1
00:34:40	Imaging	Image1
01:15:14	Weapon	WeaponRel7A

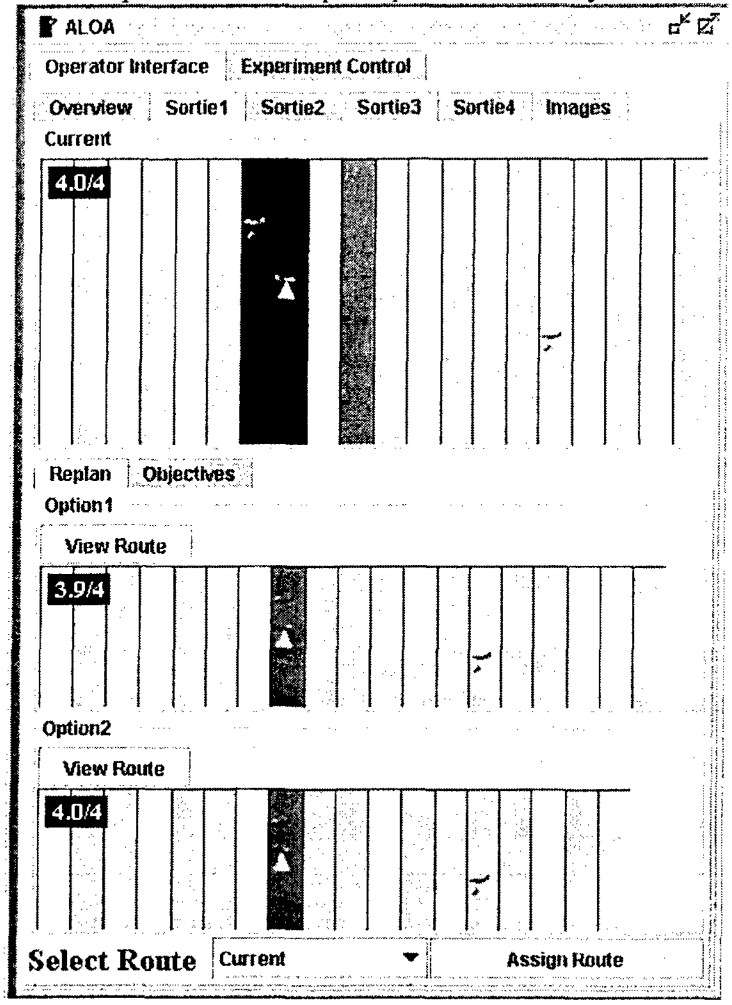
1.2.3 Sortie Replanning

When a replan event occurs, options appear in the **Replan** tab of the sortie panel. Available options depend on the autonomy level of the sortie. The possible autonomy levels are:

- Fully Automatic: no options are presented
- Automatic with Feedback: the computer makes a selection but waits for acknowledgement
- Consent: provides the operator with one optional route
- Multiple Options: provides the operator with two optional routes

- Manual: requires the operator to edit the route manually

The following is an example of the Multiple Options autonomy level.



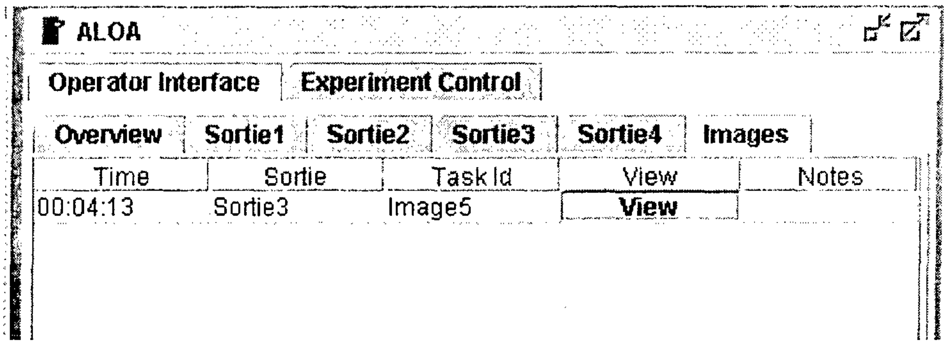
This example shows how it is possible to compare multiple routes using the FDP. Both options can avoid the SAM exposure from the current route but have slightly different EW exposure at other times along the route. It is up to the operator to decide which may be better. Also notice that the expected objectives achieved of Option1 is less than the other options, which may influence the operator.

Notice the **View Route** buttons above each optional FDP. When you click that button the route corresponding to that option displays on the map. This enables the operator to view the prospective route on the map before selecting one.

Once you have decided on a route, choose the selected option and click **Assign Route**. At that point, the sortie has been replanned. Note that you must make a selection, even if it is to keep the original route. The reason for this is because the simulation can be run at a pace faster than real time. However, in response to popup threats, the simulation will automatically change the pace to real time until all sorties have been replanned. Only when all sorties have been acknowledged will the simulation return to its original pace.

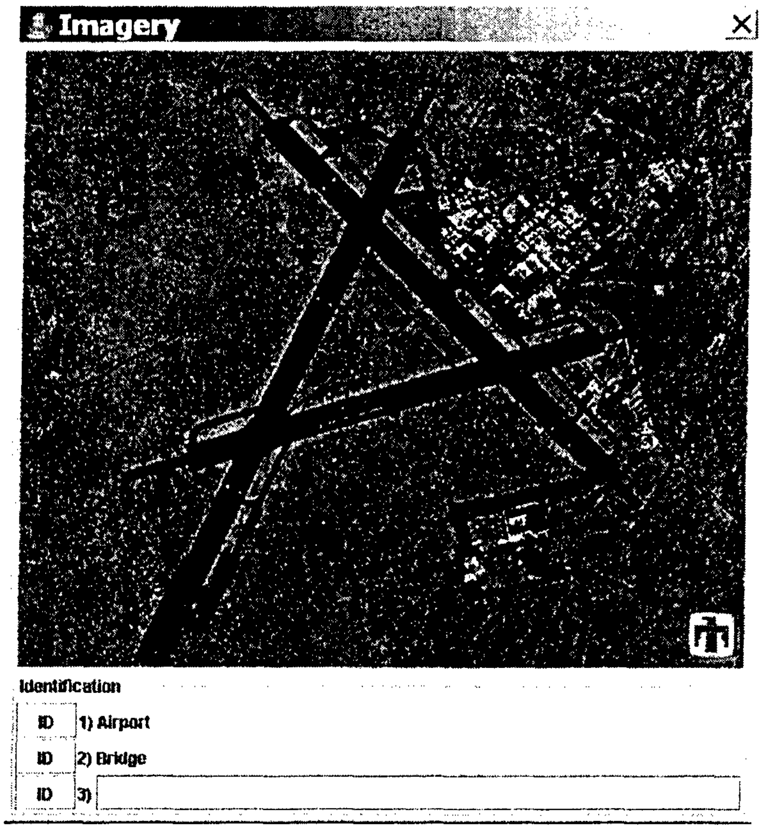
1.2.4 Images

When images become available during the simulation, a dialog appears near the bottom of the screen. This dialog will disappear by itself but can also be dismissed. To retrieve the images, select the **Images** tab. This dialog displays:



Time	Sortie	Task Id	View	Notes
00:04:13	Sortie3	Image5	View	

It indicates the time the image occurred, which sortie took the image, and the image name. To view the image, click the **View** button in the table. An image such as this displays:



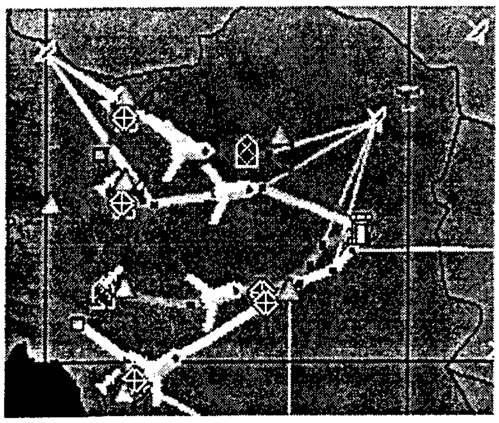
When the image appears there will be different options depending on the imaging autonomy level. Currently, the possible levels are:

- Fully Automatic: the computer identifies the image
- Multiple Options: provides the operator with multiple identification options
- Manual: the operator must identify the image without assistance

In this case, multiple options are provided. You can either select an option or manually enter another option. When you make a selection, the **Notes** field in the images table updates to reflect the selection.

1.2.5 Red Planes

Script events can cause red planes to appear on the map. Here is an example of a red plane on the map:



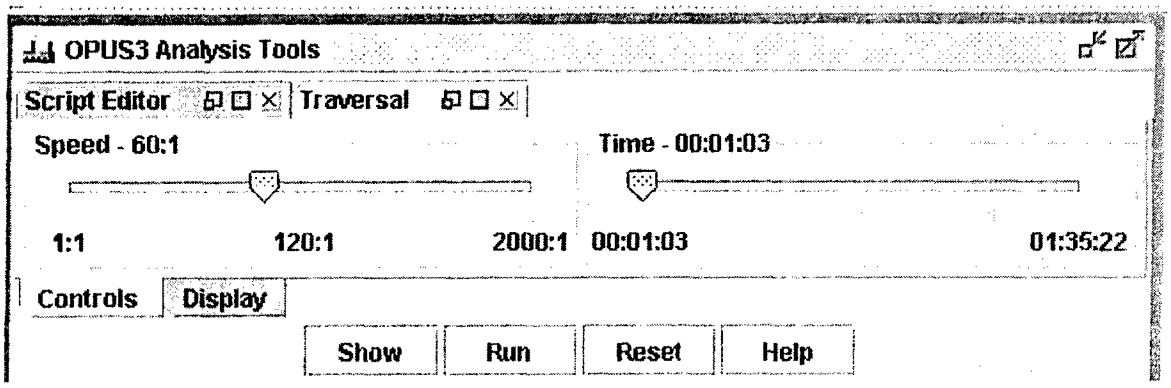
When a red plane appears, click the **Red Plane Alert** button on the overview panel. When you click the button, another dialog appears requiring you to enter a random string of numbers and letters. Once you achieve this, the red plane disappears. Note that red plane script events include a duration, which causes the red plane to disappear if you do not clear the plane during that time.

1.3 Loading a Script

You can create or load a script before starting the simulation. A companion paper describes how to create scripts. ALOA cases can be commanded to load automatically with a script (discussed in the configuration files section), or a script can be loaded after the case has been opened. Once the script is loaded, scripted events fire during the simulation. Note that in this software prototype, each case automatically loads with a script but additional scripts have been provided to load separately.

1.4 Starting the Simulation

To start the simulation, open the Analysis menu and click **Traversal**. This dialog opens:

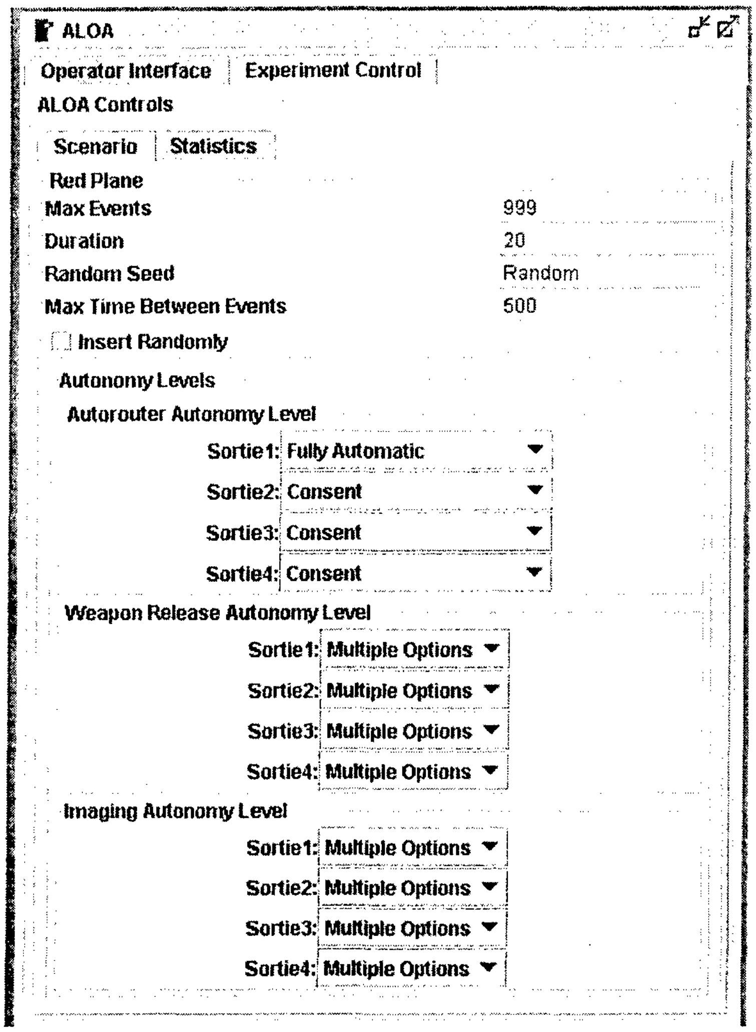


Click **Run** to start the simulation. Note that the pace of the simulation is currently set to 60:1. You can adjust the simulation's pace from this panel. During replan events, for instance when threats pop up, the pace of the simulation is automatically set to 1:1 to enable the operator time to react. However, after replanning is complete, the simulation resumes at whatever pace was selected.

Click **Pause** to pause the simulation. Also note that there are script events to blank the screen which cause the simulation to pause.

1.5 Experiment Control

In addition to a script editor, the ALOA software prototype provides some graphical control of scenario generation. Many aspects of the scenario can be controlled via OPUS data dialogs. For instance, the creation of imaging and weapon release tasks as well as vehicle performance and other data can be controlled via the OPUS data dialogs. Other ALOA experiment controls can only currently be controlled from text files. The following shows the controls that can currently be accessed:



Red Plane

Max Events	999
Duration	20
Random Seed	Random
Max Time Between Events	500

☐ Insert Randomly

Autonomy Levels

Autorouter Autonomy Level

Sortie1:	Fully Automatic
Sortie2:	Consent
Sortie3:	Consent
Sortie4:	Consent

Weapon Release Autonomy Level

Sortie1:	Multiple Options
Sortie2:	Multiple Options
Sortie3:	Multiple Options
Sortie4:	Multiple Options

Imaging Autonomy Level

Sortie1:	Multiple Options
Sortie2:	Multiple Options
Sortie3:	Multiple Options
Sortie4:	Multiple Options

You can use the controls to randomly insert red planes as well as to set autonomy levels of the sorties. To enable random red plane events, click the **Insert Randomly** check box and provide red plane control parameters. To set a sortie's autorouter, weapon releases, and imaging autonomy levels, select the appropriate level from the dropdown box.

1.6 Statistics

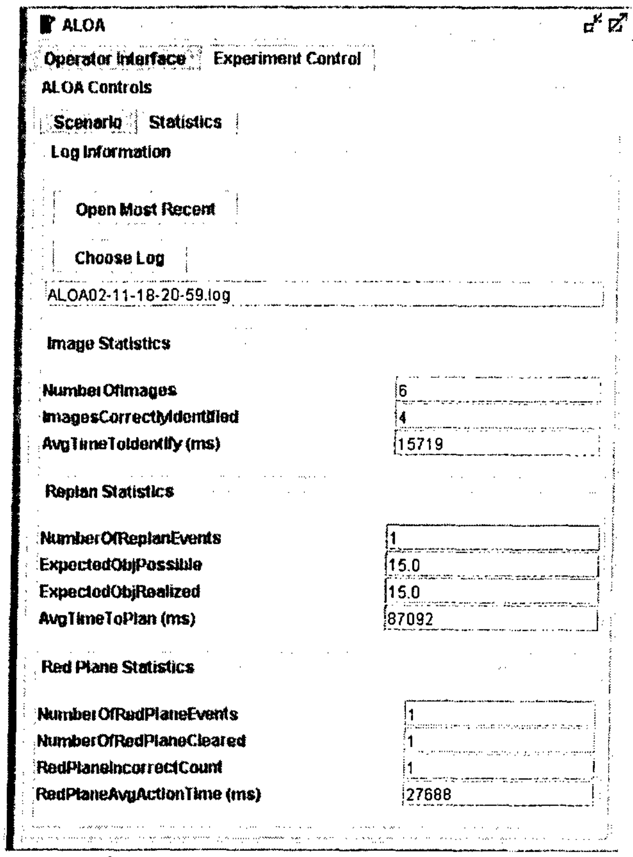
ALOA produces a log file of major events that are recorded with time stamps at the millisecond level. Events that are recorded include:

- button presses
- tab switching
- popup threats
- red plane events, and
- image identifications

Here are sample contents of a log:

```
02-15-15-06-38-251:IMAGE:Image5:AVAILABLE:Airport
02-15-15-06-43-129:GUI:DIALOG:OPERATORINTERFACE:SWITCHTAB:Images
02-15-15-06-43-848:IMAGE:Image5:VIEWED
02-15-15-06-45-709:IMAGE:Image5:IDENTIFIED:Airport
02-15-15-06-55-853:REPLAN:1:POPUP:EW:EW1-XX
02-15-15-06-55-853:REPLAN:1:ARLEVEL:Sortie1:3
02-15-15-06-55-853:REPLAN:1:ARLEVEL:Sortie2:3
02-15-15-06-55-853:REPLAN:1:ARLEVEL:Sortie3:3
02-15-15-06-55-853:REPLAN:1:ARLEVEL:Sortie4:3
02-15-15-06-56-463:REPLAN:1:FOM:Current:Sortie1:4.0/4:4.0/4
02-15-15-06-56-682:REPLAN:1:FOM:Current:Sortie2:4.0/4:4.0/4
02-15-15-06-56-901:REPLAN:1:FOM:Current:Sortie3:3.0/3:3.0/3
02-15-15-06-57-120:REPLAN:1:FOM:Current:Sortie4:4.0/4:4.0/4
02-15-15-07-00-12:GUI:DIALOG:OPERATORINTERFACE:SWITCHTAB:Sortie1
02-15-15-07-02-59:REPLAN:1:FOM:Option1:Sortie1:4.0/4:3.1/4
02-15-15-07-07-405:REPLAN:1:FOM:Option1:Sortie2:4.0/4:3.4/4
02-15-15-07-11-923:REPLAN:1:FOM:Option1:Sortie3:3.0/3:3.0/3
02-15-15-07-15-940:GUI:DIALOG:OPERATORINTERFACE:SWITCHTAB:Sortie2
02-15-15-07-16-440:GUI:DIALOG:OPERATORINTERFACE:SWITCHTAB:Sortie3
02-15-15-07-16-925:GUI:DIALOG:OPERATORINTERFACE:SWITCHTAB:Sortie4
02-15-15-07-17-175:REPLAN:1:FOM:Option1:Sortie4:4.0/4:4.0/4
02-15-15-07-18-82:GUI:DIALOG:OPERATORINTERFACE:SWITCHTAB:Sortie1
```

The software prototype provides a panel to access and display statistics from an arbitrary log file. The following shows statistics from a selected log file:



ALOA02-11-18-20-59.log

Image Statistics

NumberOfImages	6
ImagesCorrectlyIdentified	4
AvgTimeToIdentify (ms)	15719

Replan Statistics

NumberOfReplanEvents	1
ExpectedObjPossible	15.0
ExpectedObjRealized	15.0
AvgTimeToPlan (ms)	87092

Red Plane Statistics

NumberOfRedPlaneEvents	1
NumberOfRedPlaneCleared	1
RedPlaneIncorrectCount	1
RedPlaneAvgActionTime (ms)	27688

From this panel, you can access either the most recent log, which would correspond to the last run of an experiment, or you can select an arbitrary log. When a log is selected statistics are displayed. ALOA currently displays statistics regarding image identifications, replanning, and red planes.

1.7 Configuration Files

Scenario specific configuration files can be created and commanded to load when a scenario is opened. The ALOA configuration files provided with the prototype are in the

DemoCases\cases\Configs directory. The configuration files are in XML format and are self explanatory. Options that can be set from the configuration file are:

- The name of autorouter autonomy level 1-5 can be specified.
- The name of imaging autonomy level 1-3 can be specified.
- The name of weapon release autonomy level 1-3 can be specified.
- The default number of divisions for the FDP as well as threshold levels for EW, tracker, and SAM exposure can be specified.
- A script can be specified to load automatically.
- The probability to degrade expected objectives achieved can be specified.
- Images, which include a path to the image as well as automatic target recognition (ATR) options for the image, can be specified.

1.8 Sample Scenario and Script Descriptions

1.8.1 Iran.ALOA

1.8.1.1 Description

The forces under control are four UAVs. Each UAV is armed with Joint Direct Attack Munition (JDAM) standoff weapons for striking high value targets and Suppression of Enemy Air Defenses (SEAD). Weapon release objectives occur at five targets, four of which are protected by SAM shooters and SAM tracking radars. Two targets are imaged.

1.8.1.2 Scripts

- *Iran.ALOA.Script*: This script creates an early warning radar. It sets the autorouter autonomy level to Fully Automatic. Finally, it runs an unidentified vehicle test and blanks the screen near the end of the sorties.
- *Iran.ALOA.Threats*: This script updates the location of threats, after which the autorouter level of autonomy for Sortie4 is set to Automatic due to the low value of the target (Target4) and less exposure. The script adds threats, making an objective (Target7) more dangerous to accomplish.

1.8.2 Iraq.ALOA

1.8.2.1 Description

Two UCAVs are under operator control. Each UCAV is armed with JDAM standoff weapons. The UCAVs image two targets. Later, a UCAV strikes the target and the tailing UCAV images the target site for post-strike Battle Damage Assessment (BDA). After the objectives are completed, the UCAVs return to base, refueling along the route. Additionally, one UCAV follows a waypoint segment.

1.8.2.2 Scripts

- *Iran.ALOA.UCAV1*: Threats appear such that the UCAV1 sortie must be carefully monitored. A SAM shot will be fired at the UCAV1 if the sortie is not changed. The operator has little time to replan.

- *Iran.ALOA.UCAV2*: Threats change such that the UCAV2 sortie must be monitored. The UCAV1 sortie has few events, requiring less attention and allowing greater automation.

1.8.3 Iran.TwoSorties

1.8.3.1 Description

Two UAVs are under operator control, flying across Iran from one air base to another. Each UAV images a target midway through the sortie.

1.8.3.2 Scripts

- *Iran.TwoSorties*: During the sortie, three pairs of popup threats, consisting of SAM shooters and SAM tracking radars, appear. The autorouter level of autonomy is adjusted to handle the replanning time. The first pair of threats pops up near the UAVs, leaving only seconds for replanning, and the autonomy level is set to Automatic. The second set of threats occurs far away from the UAVs, allowing more than ten minutes to replan. The autonomy level is set to Manual. The third pair of threats again appears near the UAVs, but the operator is given several minutes to replan and is provided with multiple replanning options.

1.9 Included Files

1.9.1 Demo Cases

Three separate scenarios with scripts are included to demonstrate ALOA functionality. All required data and configuration files are contained in the DemoCases directory.

1.9.2 Dependent Files

ALOA is dependent on various libraries and jars, including OPUS, OpenMap, and Xerces. The dependent files as well as several configuration files are in the Dep directory in the ALOA software package.

1.9.3 Source Code

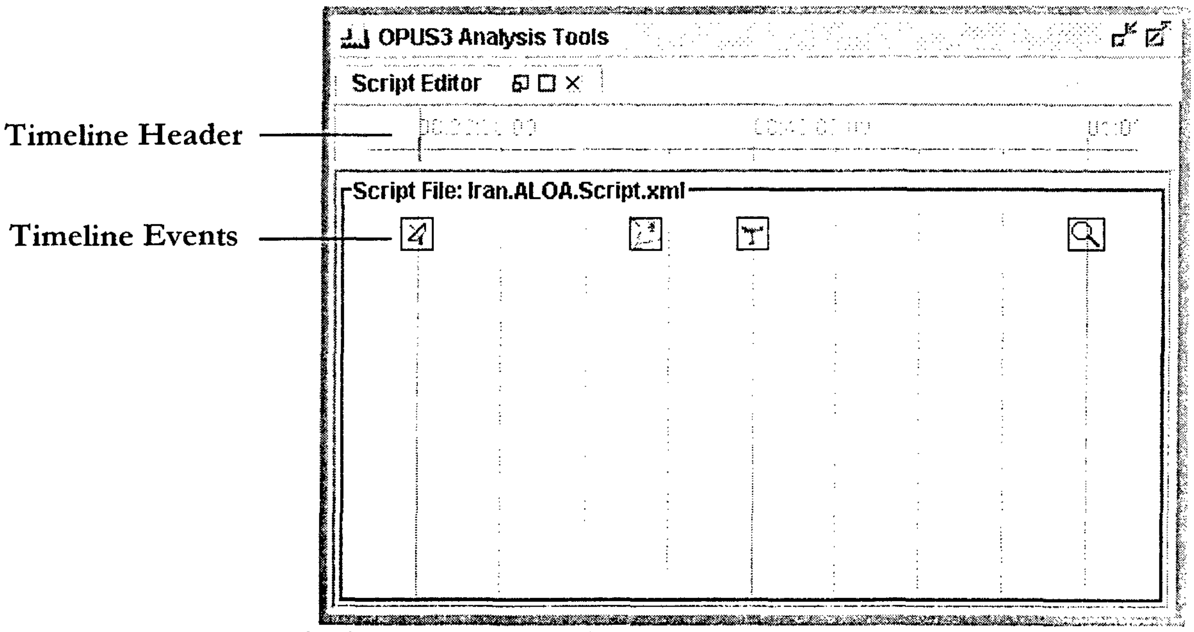
The source code for ALOA is in the Src directory in the ALOA software package.

2.0 Script Editor

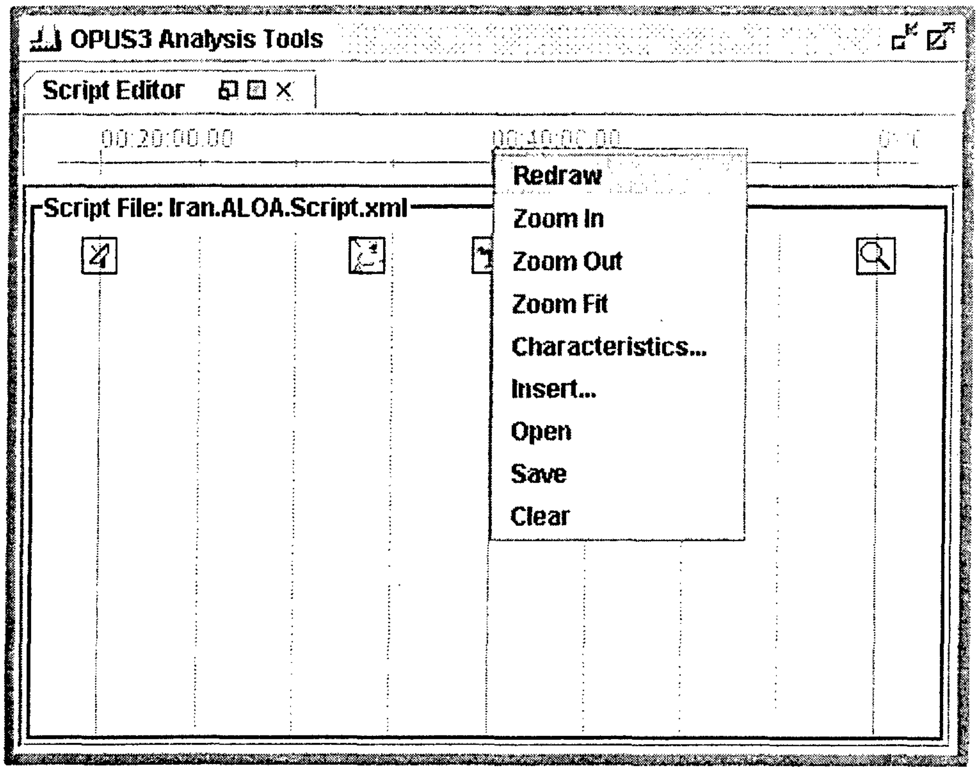
The script editor allows an experimenter to script events during ALOA simulation. This facilitates experimental setup and scenarios. See the HTML files for possible script events.

2.1 Startup

At startup, the script editor is blank, or you can load it with a predefined script file.



The **Timeline Header** indicates the current simulation time and the time interval that is displayed. The **Timeline Events** section contains script events.
Right click in the Timeline Header section for a dropdown menu, shown below.

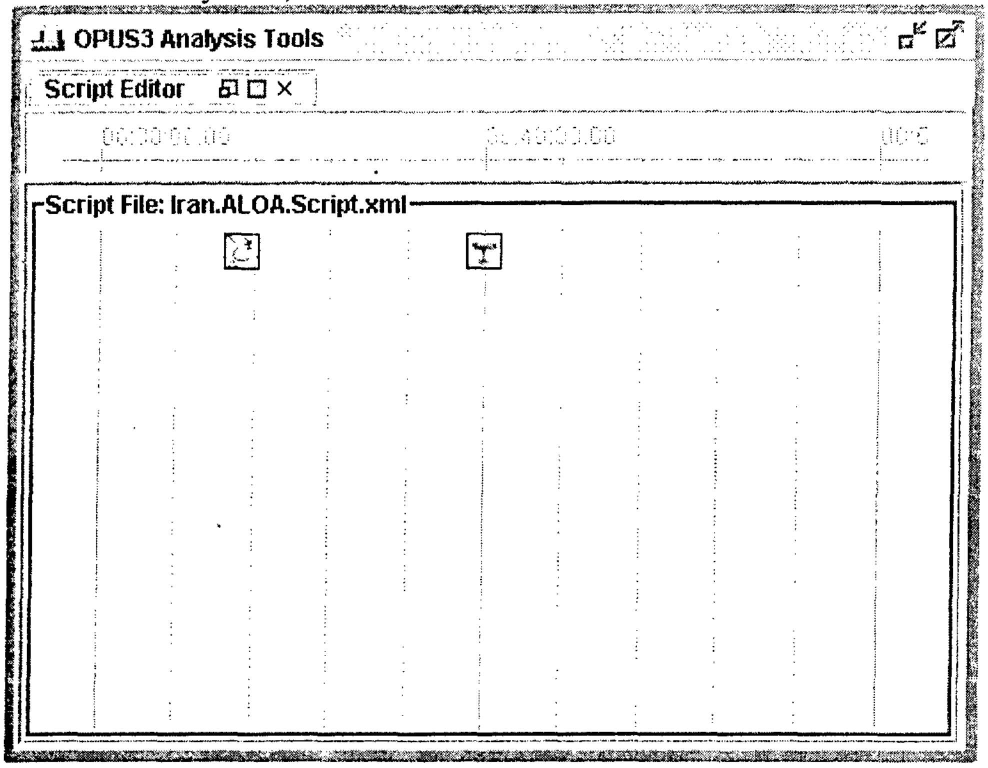


2.2 *Menu Options*

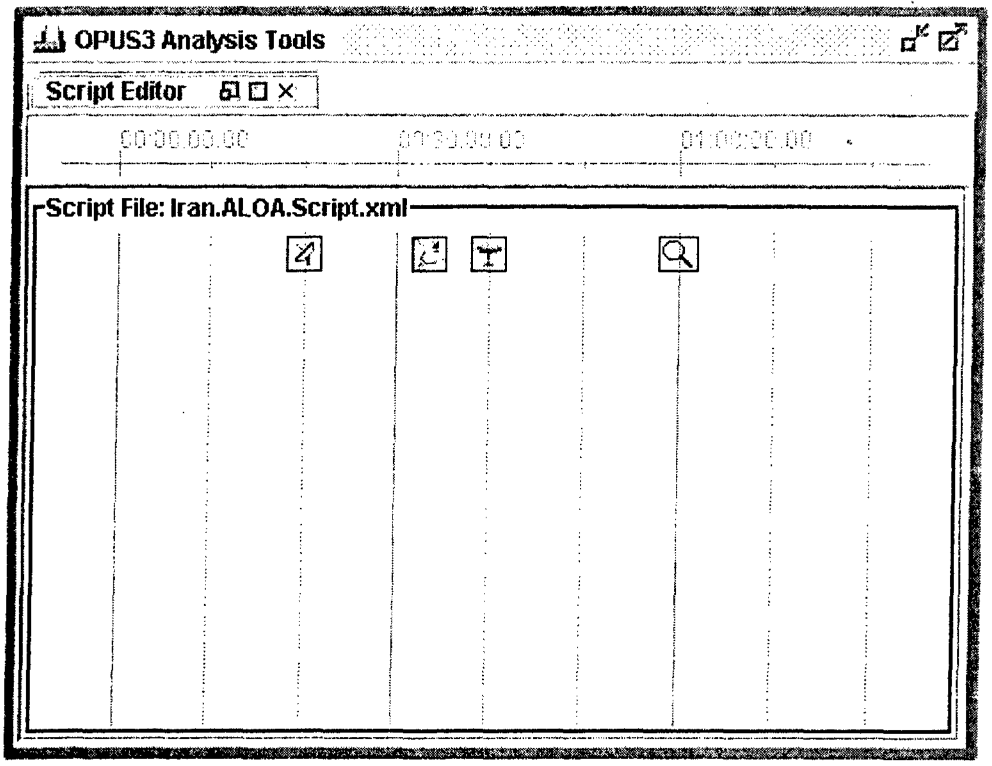
The menu options include:

- *Redraw*: refreshes the timeline.

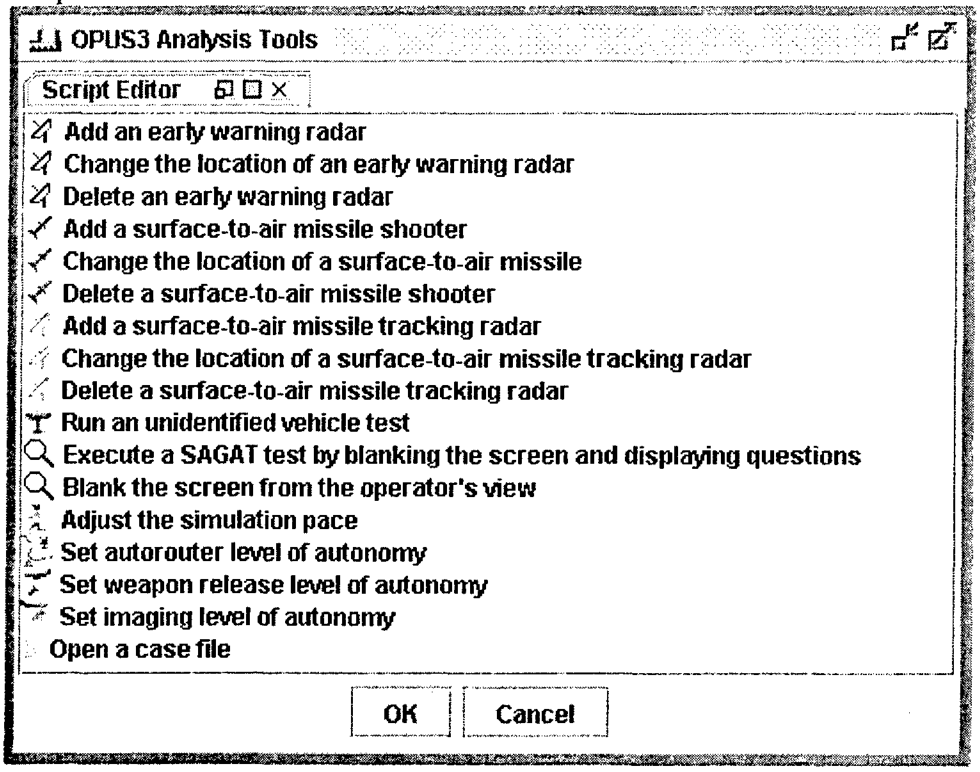
- *Zoom In*: zooms in at the location of the mouse click. This decreases the displayed time interval by 50%, as seen below.



- *Zoom Out*: zooms out at the location where the mouse was clicked. This increases the displayed time interval by 50%, as shown below.



- *Zoom Fit*: displays a time interval to show all script events.
- *Insert*: opens a menu where you can select script events. The menu below lists the script events that can be inserted.

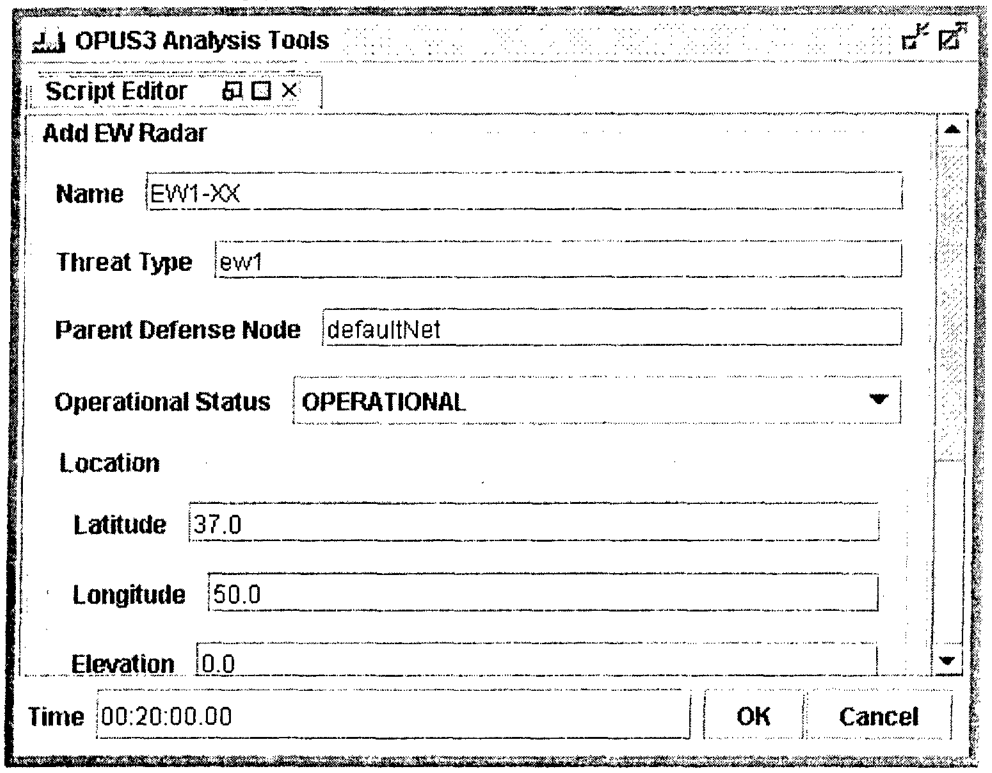


Add an early warning radar
Change the location of an early warning radar
Delete an early warning radar
Add a surface-to-air missile shooter
Change the location of a surface-to-air missile
Delete a surface-to-air missile shooter
Add a surface-to-air missile tracking radar
Change the location of a surface-to-air missile tracking radar
Delete a surface-to-air missile tracking radar
Run an unidentified vehicle test
Execute a SAGAT test by blanking the screen and displaying questions
Blank the screen from the operator's view
Adjust the simulation pace
Set autorouter level of autonomy
Set weapon release level of autonomy
Set imaging level of autonomy
Open a case file

2.3 Parameters

After selecting a script, you can enter various parameters, such as the start time of the script. It is important to note that when inserting a threat, the threat parameters ‘Threat Type’ and ‘Parent Defense Node’ must reference existing data (e.g., a SAM’s threat type references the existing threat type SA-X). Similarly, when moving or deleting a threat, the script must reference names of existing threats. Otherwise, the script has no effect.

Here is an example:



Add EW Radar

Name: EW1-XX

Threat Type: ew1

Parent Defense Node: defaultNet

Operational Status: OPERATIONAL

Location

Latitude: 37.0

Longitude: 50.0

Elevation: 0.0

Time: 00:20:00.00

Click **OK** to insert the script into the timeline. Click **Cancel** at any time to return to the Timeline Events screen.

- *Open*: loads a script file. You are prompted for a file to open.
- *Save*: saves the current script events to a file. You are prompted for a location to save the file.
- *Clear*: removes all script events.

2.4 Moving Scripts

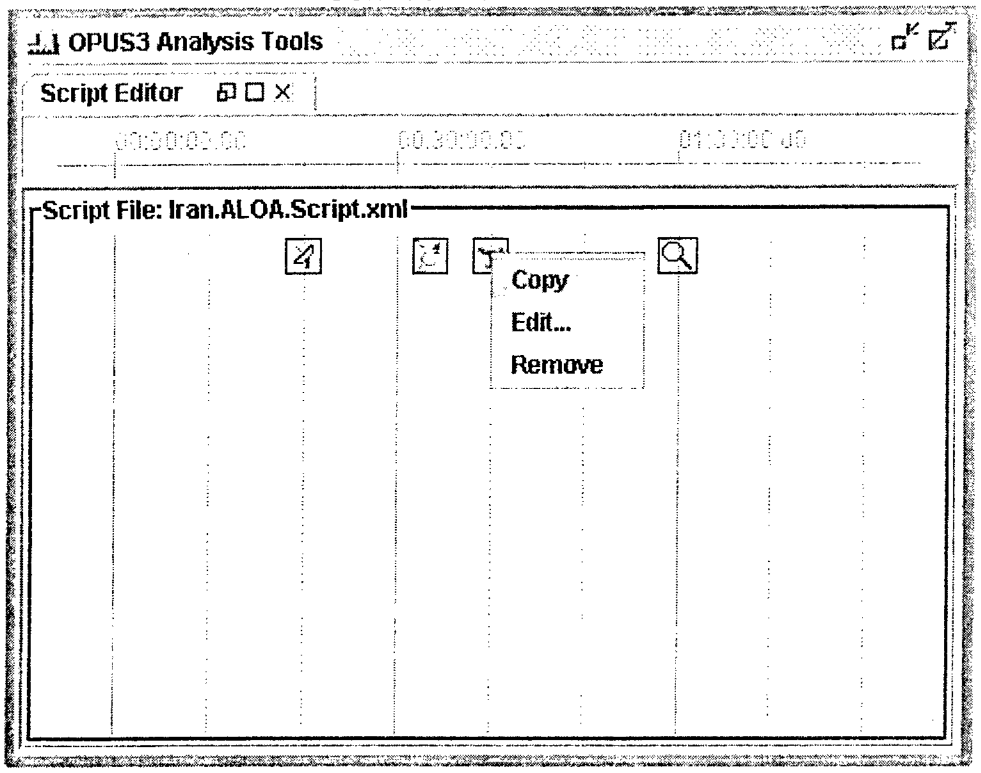
After scripts are inserted, they can be moved to different times with a mouse drag-and-drop gesture. That is, move the script by:

1. clicking on a script's icon
2. holding the mouse button down, and
3. moving the mouse to a new location.

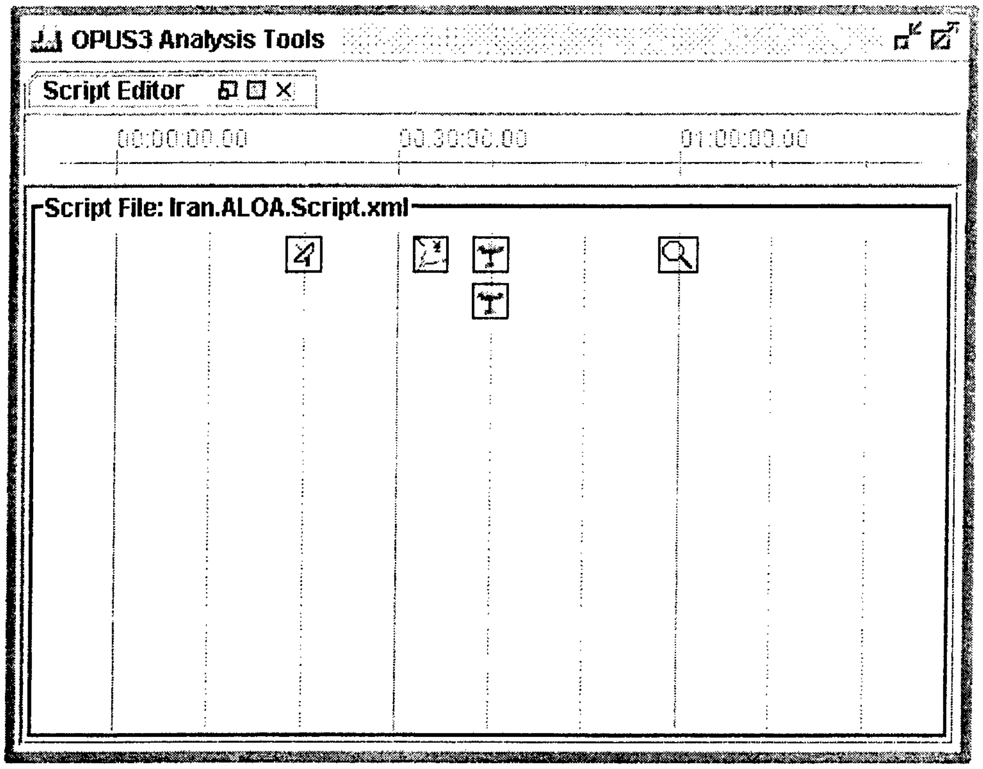
Releasing the mouse button causes the script to take place at a new start time, corresponding to the location of the mouse on the timeline.

2.5 Copy, Edit, Remove

Right click a script to open a menu with options to copy, edit, or remove the script.



Copy creates a duplicate of the script event. This is the result:



Edit enables you to modify the script's parameters and brings up the same dialog used for inserting script events. **Remove** deletes the script. You can also edit a script by double clicking the script's icon.

Any scripts assigned a time less than zero take effect immediately. For example, when a script file is loaded and it contains a script event set to Time -1 to open a scenario file, the scenario file opens immediately.